APLIKASI KRIPTOGRAFI UNTUK PENYANDIAN CITRA DENGAN MENGGUNAKAN ALGORITMA *TWOFISH*

TUGAS AKHIR

Diajukan Sebagai Salah Satu Syarat
Untuk Memperoleh Gelar Sarjana Teknik Pada
Jurusan Teknik Informatika

Oleh :

RAHMAT ELDIAN
10451025556

FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI SULTAN SYARIF KASIM RIAU
PEKANBARU
2010

APLIKASI KRIPTOGRAFI UNTUK PENYANDIAN CITRA MENGGUNAKAN ALGORITMA *TWOFISH*

RAHMAT ELDIAN

10451025556

Tanggal Sidang : 23 Juni 2010

Periode Wisuda : Juli 2010

Jurusan Teknik Informatika

Fakultas Sains dan Teknologi

Universitas Islam Negeri Sultan Syarif Kasim Riau

ABSTRAK

Pengiriman gambar melalui jaringan internet saat ini telah banyak digunakan, namun pengiriman tersebut belum tentu aman. Salah satu cara pengamanan dalam pengiriman gambar adalah dengan teknik kriptografi. Tujuan dari tugas akhir ini adalah untuk menghasilkan program aplikasi yang mampu menjaga informasi berupa *file* citra yang disampaikan agar tidak dapat dibaca atau diubah oleh pihak lain yang tidak bertanggung jawab.

Aplikasi ini dibangun menggunakan algoritma *Twofish*, algoritma ini adalah algoritma kriptografi kunci simetris berjenis *block chi*per. Algoritma *Twofish* untuk penyandian citra menggunakan 128 *bit* masukan untuk setiap 16 kali putaran secara berulang-ulang, desainnya mudah untuk dianalisa. *Twofish* merupakan algoritma kriptografi yang cepat dan kuat.

Hasil dari penelitian ini adalah suatu aplikasi yang dapat melakukan proses enkripsi dan dekripsi untuk *file* gambar dengan format bmp. Aplikasi diuji dengan berbagai *file* bmp dengan berbagai ukuran untuk mengetahui waktu yang diperlukan dan kemampuan dari aplikasi ini.

Kata kunci : Algoritma *Twofish*, Citra bmp, Dekripsi, Enkripsi, Kriptografi.

THE APPLICATION OF CRYPTOGRAPHY FOR SECRECY IMAGE BY USING TWOFISH ALGORITHM

RAHMAT ELDIAN

10451025556

Date of Final Examination : June 23th, 2010

Graduation Period : July 2010

Informatics Program

Faculty of Science and Technology

State Islamic University of Sultan Sharif Kasim Riau

ABSTRACT

Nowdays, sending a picture by using internet network is widely used, but it is not safe. Sending a picture by using cryptographic technique is one of the way to over come that problem. The aim of this research is to provide an application program which be able to keep the information in the form of image in order to make it unread by the others who are not responsible.

The application is made by using Twofish algorithm. It is a kinds of block chipper symmetric key of algorithm cryptographic. Twofish algorithm uses 128-bit input for every 16 times of round repeatedly, its design is easy to be analyzed. Twofish is fast and powerful.

The results of this research is an application of image with BMP format which can perform encryption and decryption. it is tasted by using various BMP files which various sizes to determine the time required and the capability of it.

Keywords: Algorithm Twofish, Bmp image, Cryptography, Decryption, Encryption.

DAFTAR ISI

Halaman

LEMBAR PERSETUJUAN ii
LEMBAR PENGESAHAN iii
LEMBAR HAK ATAS KEKAYAAN INTELEKTUAL iv
LEMBAR PERNYATAAN v
LEMBAR PERSEMBAHAN vi
ABSTRAK vii
ABSTRACT viii
KATA PENGANTAR ix
DAFTAR ISI xii
DAFTAR TABEL xvi
DAFTAR GAMBAR xvii
DAFTAR LAMPIRAN xix

BAB I PENDAHULUAN I-1
1.1 Latar Belakang I-1
1.2 Rumusan Masalah I-3
1.3 Batasan masalah I-4
1.4 Tujuan I-4
1.5 Sistematika Penulisan I-4

BAB II LANDASAN TEORI II-1
2.1 Keamanan Data II-1
2.2 Kriptografi II-2
2.2.1. Sejarah Kriptografi II-3
2.2.2. Algoritma Kriptografi II-6
2.2.3 Jenis Algoritma Kriptografi II-9
2.2.3.1 *Asymmetric Algorithms* II-7

2.2.3.2 *Symmetric Algorthms* II-10
2.3 Algoritma *Twofish* II-14
2.4 Kriptanalis *Twofish* II-29
2.5 Bilangan Biner II-32
2.6 *Little Endian* II-32
2.7 Citra Digital II-33

BAB III METODOLOGI PENELITIAN III-1
3.1 Perumusan masalah III-2
3.2 Studi pustaka III-2
3.3 Analisa III-3
3.4 Perancangan Program III-3
3.5 Implemantasi III-3
3.6 Pengujian III-4
3.7 Kesimpulan dan saran III-4

BAB IV ANALISIS DAN PERANCANGAN IV-1
4.1 Analisis IV-1
4.1.1 Analisis Masalah IV-1
4.1.2 Analisa Kebutuhan Sistem IV-2
4.1.2.1 Kebutuhan Masukan IV-2
4.1.2.2 Kebutuhan Fungsi atau Proses IV-2
4.1.2.3 Analisa Keluaran Proses IV-4
4.1.3 Analisa Metode IV-4
4.2 Perancangan IV-10
4.2.1 Perancangan Antar Muka Yang Akan Dibangun IV-10
4.2.2 Perancangan Antar Muka Aplikasi IV-11

BAB V IMPLEMENTASI DAN PENGUJIAN V-1
5.1 Implementasi Sistem V-1
5.1.1 Alasan Pemilihan Perangkat Lunak V-1
5.1.2 Alasan Pemilihan File V-2
5.1.3 Batasan Implementasi V-2
5.1.4 Lingkungan Implementasi V-3
5.1.5 Tampilan aplikasi V-3
5.2 Pengujian Sistem V-7
5.2.1 Lingkungan Pengujian Sistem V-8
5.2.2 Pengujian Tampilan Aplikasi V-9
5.2.2.1 Pengujian Tampilan Proses Enkripsi V-9
5.2.2.2 Pengujian Tampilan Proses Dekripsi V-12
5.2.2.3 Pengujian Tampilan Info V-14
5.2.3 Pengujian Modul V-14
5.2.3.1 Pengujian Modul Enkripsi V-14
5.2.3.2 Pengujian Modul Dekripsi V-17
5.2.4 Pengujian terhadap Jumlah Enkripsi dan Dekripsi ... V-20
5.2.5 Pengujian terhadap waktu dengan Jumlah Enkripsi yang berbeda V-20
5.2.6 Pengujian terhadap *Attack* dengan cara mencoba kemungkinan kunci V-21
5.2.5.1 Batasan Pengujian Terhadap *Attack* V-23
5.3 Kesimpulan Pengujian V-24
BAB VI PENUTUP VI-1
6.1 Kesimpulan VI-1
6.2 Saran VI-1
DAFTAR PUSTAKA
LAMPIRAN
DAFTAR RIWAYAT HIDUP

BAB I

PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi telekomunikasi dan penyimpanan data dengan menggunakan komputer memungkinkan pengiriman data jarak jauh yang relatif cepat dan murah. Dilain pihak pengiriman data jarak jauh melalui jaringan internet yang digunakan masyarakat luas *(public)* sangat memungkinkan pihak lain dapat mengakses data yang dikirimkan.

Dalam teknologi informasi, telah dan sedang dikembangkan cara-cara untuk menangkal berbagai bentuk serangan semacam itu. Salah satu cara yang ditempuh mengatasi masalah ini ialah dengan menggunakan kriptografi yang mengubah *plaintext* (bentuk file asli) menjadi *chipertext* (bentuk yang sudah tidak dimengerti) sehingga data yang dihasilkan tidak dapat dimengerti oleh pihak ketiga. *Chipertext* ini memberikan solusi pada dua masalah keamanan data, yaitu masalah privasi *(privacy)* dan keautentikan *(authentication).* Privasi mengandung arti bahwa data yang dikirim hanya dapat dimengerti informasinya oleh penerima yang sah. Sedangkan keauntentikan mencegah pihak ketiga untuk mengirimkan data yang salah atau mengubah data yang dikirimkan.

Kriptografi merupakan salah satu metode pengamanan data yang dapat digunakan untuk menjaga kerahasiaan data, keaslian data, serta keaslian pengirim.

Metode ini bertujuan agar informasi yang bersifat rahasia yang dikirim melalui telekomunikasi umum seperti LAN atau internet, tidak dapat diketahui atau dimanfaatkan oleh orang yang tidak berkepentingan atau yang tidak berhak menerimanya. Kriptografi biasanya dalam bentuk enkripsi. Proses enkripsi merupakan proses untuk meng-*encode* data dalam bentuk yang hanya dapat dibaca oleh sistem yang mempunyai kunci untuk membaca data tersebut. Hasil enkripsi disebut *cipher*. *Cipher* kemudian didekripsi dengan *device* dan kunci yang sejenis dan kembali menghasilkan *plaintext.*

Twofish adalah algoritma kriptografi modern yang dirancang dengan memperhatikan kriteria-kriteria yang diajukan oleh *National Institude of Standard Technology* (NIST), suatu badan yang menetapkan standar enkripsi kriptografi. NIST mengeluarkan produk baru yang diberi nama *Advanced Encryption Standard* (AES) untuk menggantikan DES. Salah satu kandidat AES adalah *twofish*, *twofish* mempunyai kelebihan diantara kandidiat AES yang lain. *Twofish* merupakan pengambangan dari algoritma *blowfish*.

Saat ini informasi yang bersifat penting tidak hanya dalam bentuk file teks namun penggunaan file multimedia berupa gambar yang semakin luas dimasyarakat juga tidak menutup kemungkinan untuk dirahasiakan. Citra bitmap atau yang sering disingkat dengan BMP adalah representasi dari citra grafis yang terdiri dari susunan titik yang tersimpan di memori komputer. BMP merupkan file gambar yang dapat dibuka disemua program pengolah gambar, disamping itu ukuran *byte*-nya dapat dihitung tinggi dan lebarnya dalam pixel sehingga sangat mudah untuk disandikan.

Pengiriman gambar melalui internet tidak luput dari pembajakan yang sering kali dilakukan oleh para hecker atau orang-orang yang tidak berhak. Rahasia atau pun pesan yang dikirim lewat pesan gambar akan sangat mudah untuk dilihat atau dibaca oleh orang yang tidak berhak yang pada akhirnya akan merugikan orang yang mengirimkan pesan tersebut. Untuk itu perlu dibuat suatu aplikasi untuk pengamanan data berupa gambar agar kerahasiaan dan keaslian gambar dapat terjaga dengan baik.

Dalam tugas akhir ini penulis akan membahas lebih lanjut tentang aplikasi kriptografi dengan algoritma *twofish* yang memberikan keamanan pada *file* gambar BMP-24 bit, sehingga *file* gambar tersebut dapat di enkripsi dan dekripsi menjadi data yang bersifat rahasia.

1.2 Rumusan Masalah

Sebagaimana telah dipaparkan sebelumnya pada latar belakang, maka didapatkan rumusan masalah dari tugas akhir ini, yaitu: Bagaimana membangun suatu aplikasi kriptografi yang dapat mengenkripsi dan dekripsi *file* gambar dengan menggunakan algoritma *twofish*.

1.3 Batasan Masalah

Adapun batasan masalah dalam tugas akhir ini adalah sebagai berikut :

1. Penelitian pada tugas akhir ini dibatasi pada citra dengan file bitmap-24 bit.

2. Pengujian aplikasi yang dirancang dengan mencoba kemungkinan kunci secara *random* acak.

1.4 Tujuan

Tujuan yang ingin dicapai dalam penyusunan tugas akhir ini adalah: Membangun aplikasi kriptografi untuk penyandian citra menggunakan algoritma *Twofish*

1.5 Sistematika Penulisan

Sistematika penulisan dalam penyusunan laporan tugas akhir ini adalah sebagai berikut:

BAB I PENDAHULUAN

Bab ini menjelaskan dasar-dasar dari penulisan laporan tugas akhir, yang terdiri dari latar belakang, rumusan masalah, batasan masalah, tujuan, serta sistematika penulisan laporan tugas akhir.

BAB II LANDASAN TEORI

Bab ini membahas teori-teori yang berhubungan dengan topik penelitian, meliputi keamanan data, kriptografi, algoritma *twofish* dan citra digital.

BAB III METODOLOGI PENELITIAN

Bab ini membahas tentang metodologi yang digunakan dalam penelitian dan pengembangan perangkat lunak.

BAB IV ANALISIS DAN PERANCANGAN

Pada bab ini merupakan pembahasan tentang analisis perangkat lunak, meliputi analisis, analisis masalah, analisis metode, analisis kebutuhan sistem, serta perancangan. Perancangan sistem yang terdiri dari perancangan diagram alir (*flowchart*).

BAB V IMPLEMENTASI DAN PENGUJIAN

Bab ini membahas implementasi dan pengujian yang dilakukan terhadap Aplikasi kriptografi untuk penyandian citra yang menggunakan algoritma *twofish*.

BAB VI PENUTUP

Bab ini berisi kesimpulan yang dihasikan dari pembahasan tentang Implementasi algoritma *twofish* untuk penyandian citra pada kriptografi dan saran sebagai hasil akhir dari penelitian yang telah dilakukan.

BAB II

LANDASAN TEORI

Landasan teori disusun berdasarkan teori-teori yang berhubungan dengan keamanan data, kriptografi, algoritma *Twofish* dan citra digital.

2.1 Keamanan Data

Keamanan dan kerahasiaan data saat ini menjadi isu yang sangat penting dan terus berkembang. Babarapa kasus menyangkut keamanan data saat ini menjadi suatu yang membutuhkan biaya penanganan dan pengamanan yang sedemikian besar. Sistem-sistem vital seperti sistem pertahanan, sistem perbankan, dan sistem-sistem setingkat itu membutuhkan tingkat keamanan yang sedemikian tinggi. Hal ini lebih disebabkan karena kemajuan komputer dan konsep *open system*-nya, sehingga siapapun, di manapun dan kapanpun, mempunyai kesempatan untuk mengakses kawasan-kawasan vital tersebut.

Pada garis besarnya masalah keamanan data dapat dibagi menjadi empat bidang yang saling berhubungan: kerahasiaan, keaslian, pengakuan dan kontrol integritas, yang akan dibahas nanti pada alenia berikutnya. Kerahasiaan harus dilakukan dengan menjauhkan data dari orang-orang yang tidak berhak. Keaslian berkaitan dengan siapa anda berbicara sebelum memberikan informasi yang sangat penting. Pengakuan berkaitan dengan tanda tangan digital atau sertifikasi digital. (Kristanto,2003).

2.2 Kriptografi

Keriptografi berasal dari bahasa yunani, *crypto* dan *graphia*. *Crypto* berarti *secret* (rahasia) dan *graphia* berarti *writing* (tulisan). Menurut terminologinya, kriptografi adalah ilmu dan seni untuk menjaga keamanan pesan ketika pesan dikirim dari suatu tempat ke tempat lain. (Ariyus,2008)

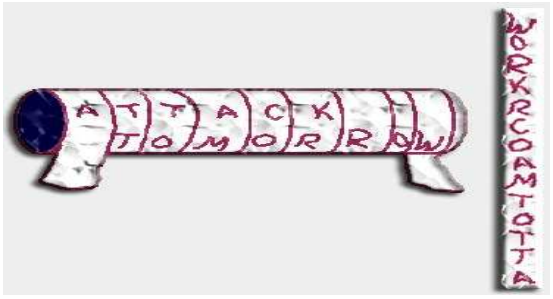
Kriptografi adalah ilmu dan seni dalam mengamankan pesan dengan cara mengubah pesan menjadi sesuatu yang tidak dapat dimerngerti oleh orang lain dengan teknik-teknik dan metode-metode tertentu. Kriptografi tidak berarti hanya memberikan keamanan informasi saja, namun lebih ke arah teknik-tekniknya. Ada empat tujuan dari ilmu kriptografi, yaitu :

1. Kerahasiaan, adalah menyediakan privasi untuk pesan dan menyimpan data dengan menyembunyikannya
2. Integritas data, berhubungan dengan penjagaan dari perubahan data secara tidak sah. Untuk menjaga integritas data, sistem harus memiliki kemampuan untuk mendeteksi manipulasi data oleh pihak-pihak yang tidak berhak, antara lain menyangkut penyisipan, penghapusan, dan pensubtitusian data lain ke dalam data yang sebenarnya
3. Autentikasi, agar penerima informasi dapat memastikan keaslian pesan, bahwa pesan itu datang dari orang yang dimintai informasi. dengan kata lain, informasi itu benar-benar datang dari orang yang dikehendaki.
4. Non-repudiasi, yang berarti dapat membuktikan bahwa dokumen memang benar datang dari orang yang dimintai informasi. Misalnya X dan X tidak menyangkalnya. (Ariyus,2008).

2.2.1. Sejarah Kriptografi

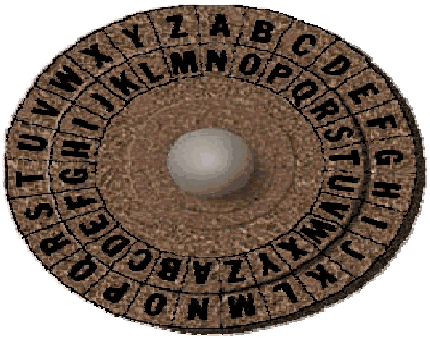
Kriptografi mempunyai sejarah yang sangat menarik dan panjang. kriptografi sudah digunakan 4000 tahun yang lalu, diperkenalkan oleh orang-orang Mesir lewat *hieroglyph.* Jenis tulisan ini bukanlah bentuk standar untuk menulis pesan. (Ariyus,2008)

Sekitar tahun 400 SM *chiper* transposisi digunakan oleh bangsa spartan di Yunani. Mereka menggunkan alat yang diberi nama *scytel* yang terdiri dari sebuah kertas panjang dari daun *papyrus* yang dililitkan pada sebuah silinder dengan diameter tertentu yang merupakan kunci penyandian. Pesan ditulis secara horizontal, baris perbaris. Bila pita dilepaskan, maka huruf-huruf di dalamnya telah tersusun secara acak membentuk pesan rahasia. Untuk membaca pesan, penerima pesan harus melilitkan kembali kertas tersebut ke silinder yang memiliki diameter sama dengan kunci penyandian



Gambar 2.1 *Scytel*

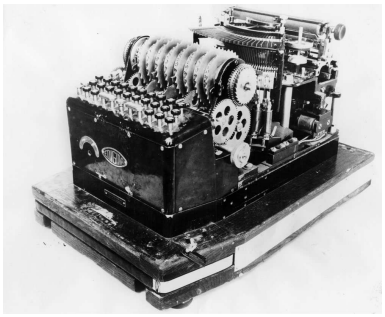
Sedangkan *chiper* subtitusi paling awal dan paling sederhana pernah dipergunakan pada zaman Romawi kuno oleh raja Yunani kuno Julius Caesar. Enkripsi dilakukan dengan menggeser karakter-karakter dalam abjad. Jika enkripsi dilakukan dengan menggeser 13 huruf ke kanan, maka huruf A disandikan dengan N, huruf B dengan O dan seterusnya.



Gambar 2.2 Ilustrasi Caesar *Chiper*

Sejarah juga mencatat bahwa kriptografi digunakan untuk tujuan keagamaan. Kalangan gereja pada masa awal agama Kristen menggunakan kriptografi untuk menjaga tulisan relijius dari gangguan otoritas politik atau budaya yang dominan pada saat itu. Mungkin yang sangat terkenal adalah *"number of the Beast"* di dalam Kitab Perjanjian Baru. Angka "666" menyatakan cara kriptografik (yaitu dienkripsi) untuk menyembunyikan pesan berbahaya (Munir, 2006).

Pada masa perang dunia ke II, tentara Nazi Jerman menggunakan mesin *Enigma* atau juga disebut dengan mesin *rotor* yang digunakan Hitler untuk mengirimkan pesan kepada tentaranya di medan perang. Masin yang menggunakan beberapa buah *rotor* (roda berputar) ini melakukan enkripsi dengan cara yang cukup rumit. Namun, *Enigma chiper* berhasil dipecahkan oleh tentara sekutu yang merupakan faktor sehingga memperpendek perang dunia ke II. Setelah Jerman mengetahui bahwa *enigma* dapat dipecahkan, maka enigma mengalami beberapa kali perubahan. *Enigma* yang digunakan Jerman dapat mengenkripsi suatu pesan sehingga mempunyai 15×1018 kemungkinan untuk dapat mendekripsi pesan.(Munir. 2006)



Gambar 2.3. Mesin Enigma

Perkembangan komputer dan sistem komunikasi pada tahun 1960-an mengakibatkan munculnya kebutuhan pihak swasta akan alat untuk melindungi informasi dalam bentuk digital dan untuk menyediakan layanan keamanan informasi. Kriptografi digital dimulai pada tahun 1970 atas usaha *Feistel* dari IBM dan memuncak pada tahun 1977 dengan diadopsinya sistem kriptografi DES (*Data Encryption Standard*) oleh *U.S. Federal Information Processing Standard* untuk mengenkripsi informasi rahasia. DES merupakan mekanisme kriptografi yang paling terkenal dalam sejarah dan tetap menjadi standar pengamanan data elektronik komersial pada kebanyakan institusi keuangan di seluruh dunia.

Sampai pada akhir Perang Dunia I, kriptografi merupakan disiplin ilmu matematika yang spesial. Penelitian dalam bidang ini tidak pernah sampai kepada umum sehingga tidaklah mengherankan kalau banyak orang tidak mengetahui keberadaan ataupun manfaat darinya.

Pengembangan paling mengejutkan dalam sejarah kriptografi terjadi pada 1976 saat Diffie dan Hellman mempublikasikan *"New Directions in Cryptography*". Tulisan ini memperkenalkan konsep revolusioner kriptografi kunci publik dan juga memberikan metode baru untuk pertukaran kunci, keamanan yang berdasar pada kekuatan masalah algoritma diskrit. Meskipun

Diffie dan Hellman tidak memiliki realisasi praktis pada ide enkripsi kunci publik saat itu, idenya sangat jelas dan menumbuhkan ketertarikan yang luas pada komunitas kriptografi.

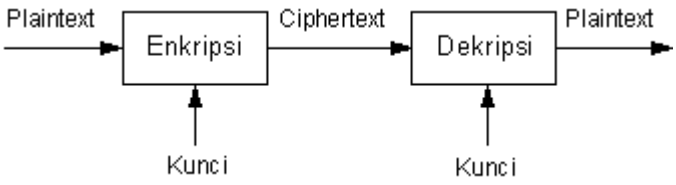
Algoritma kriptografi modern tidak lagi mengandalkan keamanannya pada kerahasiaan algoritma tetapi kerahasiaan kunci. *Plaintext* yang sama bila disandikan dengan kunci yang berbeda akan menghasilkan *chipertext* yang berbeda pula. Dengan demikian algoritma kriptografi dapat bersifat umum dan boleh diketahui oleh siapa saja, akan tetapi tanpa pengetahuan tentang kunci, data tersandi tetap saja tidak dapat terpecahkan.

2.2.2 Algoritma Kriptografi

Algoritma kriptografi terdiri dari tiga aspek dasar, yaitu:

1. Enkripsi merupakan hal yang sangat penting dalam kriptografi, merupakan pengamanan data yang dikirimkan agar terjaga kerahasiaannya. Pesan asli disebut *plaintext*, yang di ubah menjadi kode-kode yang tidak dimengerti atau disebut *chipertext*.
2. Dekripsi merupakan kebalikan dari proses enkripsi. Pesan yang enkripsi dikembalikan ke bentuk asalnya atau dengan kata lain proses membalikkan *chipertext* menjadi *plaintext*.
3. Kunci yang dimaksud disini adalah kunci yang dipakai untuk melakukan enkripsi dan dekripsi. Kunci terbagi menjadi dua bagian, kunci rahasia (*private key)* dan kunci umum *(Public key).*

Illustrasi untuk menggambarkan kondisi kriptografi dapat dilihat pada gambar di bawah:



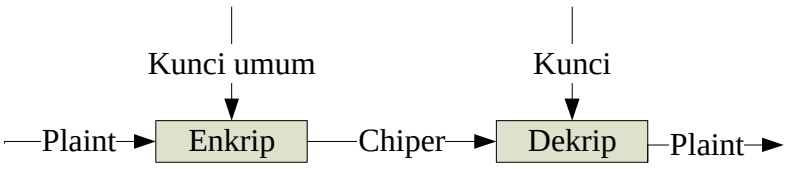
Gambar 2.4 Proses Enkripsi dan Dekripsi

2.2.3 Jenis Algoritma Kriptografi

Berdasarkan kunci yang dipakai, algoritma kriptografi dapat dibedakan atas dua golongan, yaitu :

2.2.3.1 *Asymmetric Algorithms*

Asymmetric Algorithms adalah algoritma yang menggunakan kunci yang berbeda untuk proses enkripsi dan dekripsinya. Algoritma ini disebut juga algoritma kunci umum (*public key algorithm*) karena kunci untuk enkripsi dibuat umum (*public key*) atau dapat diketahui oleh setiap orang, tapi kunci untuk dekripsi hanya diketahui oleh orang yang berwenang mengetahui data yang disandikan atau sering disebut kunci pribadi (*private key*). Proses enkripsi-dekripsi algoritma asimetris dapat dilihat pada gambar dibawah ini :



Gambar 2.5 Proses Enkripsi dan Dekripsi Algoritma Asimetris

Pada algoritma *public key* ini, semua orang dapat mengenkripsi data dengan memakai *public key* penerima yang telah diketahui secara umum. Akan

tetapi data yang telah terenkripsi tersebut hanya dapat didekripsi dengan menggunakan *private key* yang hanya diketahui oleh penerima. beberapa contoh algoritma kunci publik antara lain

1. DSA

Pada bulan Agustus 1991, NIST (*The National Institute of Standard and Technology*) mengumumkan algoritma sidik dijital yang disebut *Digital Signature Algorithm* (DSA). DSA dijadikan sebagai standar dari *Digital Signature Standard* (DSS).

DSA tidak dapat digunakan untuk enkripsi. DSA mempunyai dua fungsi utama:

1. Pembentukan sidik digital (*signature generation*)
2. Pemeriksaan keabsahan sidik digital (*signature verification*).

DSA menggunakan dua buah kunci, yaitu kunci publik dan kunci rahasia. Pembentukan sidik digital menggunakan kunci rahasia pengirim, sedangkan verifikasi sidik digital menggunakan kunci publik pengirim. DSA menggunakan fungsi *hash* SHA (*Secure Hash Algorithm*) untuk mengubah pesan menjadi *message digest* yang berukuran 160 *bit*.

2. RSA

Algoritma RSA dibuat oleh 3 orang peneliti dari MIT (*Massachussets Institute of Technology*) pada tahun 1976, yaitu: Ron (R)ivest, Adi (S)hamir, dan Leonard (A)dleman.

RSA yang menggunakan algoritma asimetris mempunyai dua kunci yang berbeda, disebut pasangan kunci (*key pair*) untuk proses enkripsi dan dekripsi.

Kunci-kunci yang ada pada pasangan kunci mempunyai hubungan secara matematis, tetapi tidak dapat dilihat secara komputasi untuk mendeduksi kunci yang satu ke pasangannya. Algoritma ini disebut kunci publik, karena kunci enkripsi dapat disebarkan. Orang-orang dapat menggunakan kunci publik ini, tapi hanya orang yang mempunyai *private key* sajalah yang bisa mendekripsi data tersebut.

Dalam pemakaian sistem penyandian RSA dalam kehidupan sehari-hari adalah *signature* atau tanda tangan *digital* dalam surat elektronik dan untuk autentikasi sebuah data. Untuk meyakinkan penerima surat elektronik yang ditandatangani, diperlukan pembuktian bahwa surat elektronik tersebut memang berasal dari sipengirim.

RSA dalam prosesnya tergolong lambat dibandingkan dengan metode enkripsi yang popular baik metode yang menggunakan kunci simetris, dimana metode lain lebih cepat.

3. LUC

Algoritma LUC merupakan algoritma kriptografi kunci public yang dikembangkan oleh Peter J. Smith dari New Zealand pada tahun 1993. Metode LUC ini dirancang oleh Peter J. Smith setelah ia berhasil meneliti dan melihat kelemahan dari metode RSA. Metode LUC ini menggunakan fungsi Lucas yang dapat menutupi kelemahan metode RSA yang menggunakan fungsi pangkat. Kemungkinan untuk menjebol RSA menjadi ada karena masalah pangkat tersebut. Fungsi Lucas ini dapat mencegah kemungkinan tersebut.

Algoritma LUC sendiri hampir sama dengan metode RSA hanya saja fungsi pangkat pada metode RSA diganti dengan fungsi Lucas pada metode LUC. Metode LUC ini menggunakan dua buah bilangan prima besar, p dan q untuk menghasilkan pasangan kunci publik, e dan n, dimana n = p * q, serta *private key*, d. Sedangkan nilai p dan q sendiri tidak digunakan dalam algoritma ini.

Algoritma ini juga menggunakan Extended Euclidean algorithm untuk menghasilkan nilai kunci dekripsi d dari kunci enkripsi e. Nilai kunci enkripsi e merupakan sebuah bilangan acak yang dibangkitkan sedemikian rupa sehingga harus relatif prima terhadap p – 1, q – 1, p + 1 dan q + 1. Sedangkan panjang *plaintext* harus lebih kecil daripada n.

2.2.3.2 *Symmetric Algorithms*

Algoritma kriptografi simetris atau disebut juga algoritma kriptografi konvensioanl adalah algoritma yang menggunakan kunci untuk proses enkripsi sama dengan kunci untuk proses dekripsi.

Algoritma kriptografi simetri dapat dikelompokkan menjadi dua kategori, yaitu:

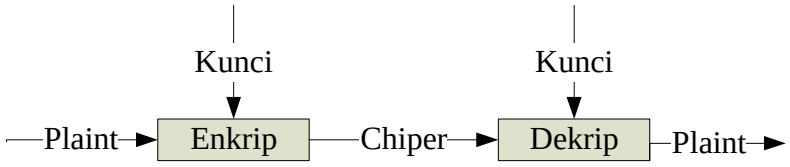
1. ***Chiper* aliran (s*tream chiper*)**

Algoritma kriptografi beroperasi pada plainteks/chiperteks dalam bentuk *bit* tunggal, pada rangkaian *bit* ini dienkripsikan dan didekripsikan *bit* per *bit*.

2. ***Chiper* blok (*block chiper*)**

Algoritma kriptografi beroperasi pada plainteks/chiperteks dalam bentuk blok *bit*, yang dalam hal ini rangkaian *bit* dibagi menjadi blok-blok *bit* yang panjangnya sudah ditentukan sebelumnya.

Proses enkripsi dan dekripsi algoritma kriptografi simetris dapat dilihat pada gambar dibawah ini :



Gambar 2.6 Proses Enkripsi dan Dekripsi Algoritma Simetris

1. Algoritma DES *(Data Encryption Standard)*

Algoritma DES merupakan kunci simetris, dikembangkan di IBM dibawah kepemimpinan W.L. Tuchman pada tahun 1972. Algoritma ini didasarkan pada algoritma LUCIFER yang dibuat oleh Horst Feistel. Algoritma DES telah mendapat persetujuan dari *National Bureau of Standard* (NBS) setelah penilaian kekuatannya oleh *National Security Agency* (NSA) Amerika Serikat

DES beroperasi pada ukuran blok 64 *bit*. DES mengenkripsikan 64 *bit* plainteks menjadi 64 *bit* chiperteks dengan menggunakan 56 *bit* kunci internal (*internal key*) atau upa kunci (*subkey*). Kunci internal dibangkitkan dari kunci eksternal (*external key*) dengan panjang kunci eksternal DES hanya 64 bit atau 8 karakter, itupun yang dipakai hanya 56 bit. Pada rancangan awal, panjang kunci yang diusulkan IBM adalah 128 bit, tetapi atas permintaan NSA, panjang kunci diperkecil menjadi 56 bit, maka DES memiliki kunci yang lemah dan desain struktur internal DES dimana bagian subsitusinya (S-box) masih dirahasiakan.

2. Algoritma IDEA

Algoritma IDEA (*International Data Encryption Algorithm*) merupakan sebuah algoritma kunci simetrik (*Symmetric Algorithms*). IDEA muncul pertama

kali pada tahun 1990 yang dikembangkan oleh ilmuwan Xueijia Lai dan James L Massey.

IDEA mengenkripsi *plaintext* menjadi chiperteks yang lemah hanya mengalami 8 putaran. Algoritma ini membagi *plaintext* yang akan dienkripsi menjadi 4 blok, masing-masing terdiri dari 16 *bit*, tetapi IDEA lebih meningkatkan proses keamanan kunci, dimana 52 upa kunci (sub-keys) yang terdiri dari 16 *bit* dibangkitkan dari kunci utama (master key) yang terdiri 128 *bit*. Lalu pada setiap putarannya digunakan 6 kunci. Setelah itu dilakukan transformasi final dengan 4 kunci untuk membalikkan posisi ke operasi dasar. Algoritma IDEA mempunyai paten untuk mencegah pembajakan dan kejahatan.

3. Algoritma *Blowfish*

Blowfish merupakan sebuah algoritma kunci simetrik (*symmetric Algorithms) chiper* blok yang dirancang pada tahun 1993 oleh Bruce Schneier. Pada saat itu banyak sekali rancangan algoritma yang ditawarkan, namun hampir semua terhalang oleh paten atau kerahasiaan pemerintah Amerika. Schneier menyatakan bahwa *blowfish* bebas paten dan akan berada pada domain publik. Dengan pernyataan Schneier tersebut *blowfish* telah mendapatkan tempat di dunia kriptografi, khususnya bagi masyarakat yang membutuhkan algoritma kriptografi yang cepat, kuat, dan tidak terhalang oleh lisensi.

Keberhasilan *blowfish* dalam menembus pasar telah terbukti dengan diadopsinya *blowfish* sebagai *Open Cryptography Interface* (OCI) pada *kernel Linux versi* 2.5 keatas. Dengan diadopsinya *blowfish*, maka telah menyatakan

bahwa dunia *open source* menganggap *blowfish* adalah salah satu algoritma yang terbaik

4. Algoritma *Rinjdael*

Seperti pada *DES*, *Rijndael* menggunakan substitusi dan permutasi, dan sejumlah putaran (*cipher* berulang) – setiap putaran mengunakan kunci internal yang berbeda (kunci setiap putaran disebut *round key*). Tetapi tidak seperti *DES* yang berorientasi bit, *Rijndael* beroperasi dalam orientasi *byte* (untuk memangkuskan implementasi algoritma ke dalam *software* dan *hardware*). Garis besar Algoritma *Rijndael* yang beroperasi pada blok 128-bit dengan kunci 128-bit adalah sebagai berikut (di luar proses pembangkitan *round key*):

1. *AddRoundKey*: melakukan *XOR* antara *state* awal (plainteks) dengan *cipher key*. Tahap ini disebut juga *initial round*.
2. Putaran sebanyak *Nr* – 1 kali. Proses yang dilakukan pada setiap putaran adalah:

   a. *SubBytes*: substitusi *byte* dengan menggunakan table substitusi (*S-box*).

   b. *ShiftRows*: pergeseran baris-baris *array state* secara *wrapping*.

   c. *MixColumns*: mengacak data di masing-masing kolom *array state.*

   d. *AddRoundKey*: melakukan XOR antara *state* sekarang *round key*.
3. *Final round*: proses untuk putaran terakhir:

   a. SubBytes

   b. ShiftRows

   c. AddRoundKey

5. Algoritma *Twofish*

Twofish merupakan algoritma yang beroperasi dalam mode *block*. Algoritma *Twofish* sendiri merupakan pengembangan dari algoritma *Blowfish*. Perancangan *Twofish* dilakukan dengan memperhatikan kriteria-kriteria yang diajukan *National Institute of Standards and Technology* (NIST) untuk kompetisi *Advanced Encryption Standard* (AES). Tujuan dari perancangan *Twofish* yang selaras dengan kriteria NIST untuk AES adalah sebagai berikut:

1. Merupakan *cipher* blok dengan kunci simetri dan blok sepanjang 128 bit.
2. Panjang kunci yang digunakan adalah 128 bit, 192 bit. Dan 256 bit.
3. Tidak mempunyai kunci lemah.
4. Efisiensi algoritma, baik pada Intel Pentium Pro dan perangkat lunak lainnya dan platform perangkat keras.
5. Rancangan yang *fleksibel*. Rancangan yang *fleksibel* ini dapat diartikan misalnya dapat menerima panjang kunci tambahan, dapat diterapkan pada platform dan aplikasi yang sangat variatif, serta cocok untuk *cipher* aliran, fungsi hash, dan MAC.
6. Rancangan yang sederhana agar memudahkan proses analisis dan implementasi algoritma.

Algoritma *Twofish*

Twofish menggunakan sebuah Struktur *Feistel-like 16-round* dengan tambahan *whitening* pada masukan dan keluaran. Satu-satunya unsur *non-Feistel* adalah 1-bit rotasi. Perputaran dapat dipindah ke dalam fungsi F untuk membuat suatu struktur Feistel murni, tapi memerlukan suatu tambahan perputaran kata-

kata yang tepat sebelum langkah keluaran whitening. *Plaintext* dipecah menjadi empat kata 32-bit. Pada langkah *whitening* masukan terdapat XOR dengan empat kata kunci. Ini diikuti oleh enambelas putaran. Pada setiap putaran, dua kata-kata pada sisi kiri digunakan sebagai masukan kepada fungsi g (Salah satu darinya diputar 8 bit pertama). Fungsi g terdiri dari empat *byte-wide S-Box key-dependent*, yang diikuti oleh suatu langkah pencampuran linier berdasar pada suatu matriks *MDS*. Hasil kedua fungsi g dikombinasikan menggunakan suatu *Pseudo Hadamard Transform (PHT)*, dan ditambahkan dua kata kunci. Kedua hasil ini kemudian di-*XOR* ke dalam kata-kata pada sisi kanan (salah satunya diputar ke kanan 1 bit pertama, yang lainnya diputar ke kanan setelahnya). Yang kiri dan kanan dibelah dua kemudian ditukar untuk putaran yang berikutnya, pertukaran yang terakhir adalah dibalik, dan yang empat kata di-*XOR* dengan lebih dari empat kata kunci untuk menghasilkan ciphertext. Secara formal, 16 *byte plaintext* p_0,… p_{15} yang yang pertama dipecah menjadi 4 kata $P_{0,...}P_3$ dari 32 bit masing-masing menggunakan konvensi *little-endian.*

NIST menggunakan suatu *block cipher*. *Block cipher* dapat digunakan untuk merancang aliran *cipher* dengan berbagai sinkronisasi untuk properti-properti kesalahan yang luas, sebuah cara fungsi *hash*, kode pengesahan pesan, dan *pseudo-random generator* . Oleh karenanya dikatakan sebagai kuda beban (*workhorse*) dari ilmu kriptograpi modren. NIST menetapkan beberapa kriteria disain lain: kunci yang lebih panjang, ukuran blok lebih luas, lebih cepat, dan *exibilitas* lebih besar. Selagi tidak ada algoritma tunggal yang dapat dioptimalkan untuk semua kebutuhan, NIST menjadikan *AES* sebagai standar algoritma simetris

untuk dekade selanjutnya. *Twofish* mengacu pada proses seleksi *AES*. Hal ini mempertemukan semua yang dibutuhkan NIST untuk kriteria 128 bit blok; 128, 192, dan 256-bit kunci; efisien pada berbagai *platform*; dan lain-lain serta beberapa kebutuhan syarat disain sebagaimana halnya kriptografi.

Blok yang membangun *Twofish* seperti dibawah ini:

1. **Jaringan Feistel**

Hampir semua algoritma blok kode bekerja dalam model jaringan Feistel. Jaringan Feistel ditemukan oleh Horst Feistel desainnya tentang Lucifer, dan dipopulerkan untuk DES. Jarimgan Fesitel adalah metode umum untuk mentransformasikan fungsi apapun (biasa disebut dengan fungsi F) ke dalam permutasi. Beberapa algoritma kriptografi lain yang menggunakan jaringan Feistel adalah LOKI, GOST, FEAL, *Blowfish*, Khufu Khafre dan RC5. Model jaringan Feistel bersifat *reversible*, untuk proses enkripsi dan dekripsi, sehingga tidak perlu membuat algoritma baru untuk mendekripsi teks-kode menjadi teks-asli.

Bagian yang penting dari jaringan Feistel adalah fungsi F. Pemetaan string masukan menjadi string keluaran berdasarkan kunci yang digunakan. Fungsi F selalu tidak linier dan mungkin tidak subjektif;

$$F\ L:(0.1)^{n/2}\times(0.1)^{N}\ (0.1)^{n/2} \quad \text{.....(2.1)}$$

Dimana n adalah ukuran blok dari jaringan Feistel dan F adalah fungsi yang menerima n/2 bit dari blok dan N bit kunci sebagai masukan dan menghasilkan n/2 bit keluaran. Dalam setiap putaran, blok sumber adalah masukkan fungsi F dan keluaran dari fungsi F di-XOR-kan dengan blok target, lalu dua blok ini dipertukarkan sebelum masuk ke putaran berikutnya. Ide yang digunakan adalah

untuk mengambil fungsi F, yang mungkin merupakan algorima enkripsi yang lemah jika berdiri sendiri, dan melakukan perulangan untuk membuat algoritma enkripsi yang kuat. Dua putaran pada jaringan Feistel disebut satu *cycle*, dan dalam satu cycle setiap bit dari blok teks telah dimodifikasi sekali. *Twofish* merupakan jaringan Feistel 16-putaran, dengan fungsi F bijektif.(Ariyus, 2008)

2. **Kotak-S (S-boxes)**

Kotak-S adalah matriks yang berisi subtitusi non-linier yang memetakan satu atau lebih bit dengan satu atau lebih bit lain dan digunakan dibanyak blok kode. Kotak-S memiliki ukuran masukan dan ukuran keluaran yang bervariasi. Ada empat pendekatan yang digunakan dalam mengisi kotak-S: dipilih secara acak, dipilih secara acak lalu diuji, dibuat orang, dihitung secara matematis. Kotak-S pertama digunakan di Lucifer, lalu DES dan diikuti banyak algoritma enkripsi yang lain. *Twofish* menggunakan empat buah 8X8 bit blok-S yang berbeda, bijektif dan bergantung pada kunci. Kotak-S bit dibuat menggunakan 8X8 bit permutasi dan material kunci.

3. **MDS Matrices**

Kode MDS (*Maximum Distance Separable*) pada sebuah field adalah pemetaan linier dari $x + y$ elemen, dengan ketentuan bahwa jumlah minimum dari elemen bukan nol pada setiap vektor bukan nol paling sedikit $y + 1$. Dengan kata lain, jumlah elemen yang berbeda diantara dua vektor berbeda yang dihasilkan oleh pemetaan MDS paling sedikit $y + 1$. Dapat dibuktikan dengan mudah bahwa tidak ada pemetaan yang dapat memiliki jarak pisah yang lebih besar diantara dua vektor yang berbeda, sehingga disebut jarak pisah maksimum. (*maximum distance*

separable). Pemetaan MDS dapat direpresentasikan dengan sebuah matriks MDS yang terdiri dari x X y elemen. Kode perbaikan kesalahan Reed-Solomon (RS) adalah MDS. Kondisi yang diperlukan untuk sebuah matriks x X y untuk menjadi MDS adalah semua kemungkinan sub-matriks kotak yang diperoleh dengan membuang kolom atau baris, adalah tidak singular.

Serge Vaudenay pertama kali mengajukan matriks MDS sebagai elemen desain kode. Shark dan Squere menggunakan metriks MDS, meskipun konstruksinya pertama kali ditemukan di kode Manta yang tidak dipublikasikan. *Twofish* menggunakan sebuah matriks MDS 4x4 pada GF(2^8).

4. **Transformasi Pseudo-Hadamard (PHT)**

Adalah sebuah operasi pencampuran sederhana yang berjalan secara cepat dalam perangkat lunak PHT 32-bit dengan dua masukkan didefenisi sebagai berikut:

$$a' = a + b \bmod 2^{32}$$
$$b' = a + 2b \bmod 2^{32} \quad \text{.....(2.2)}$$

Twofish menggunakan PHT 32-bit untuk mengubah keluaran dari fungsi *g*-nya.

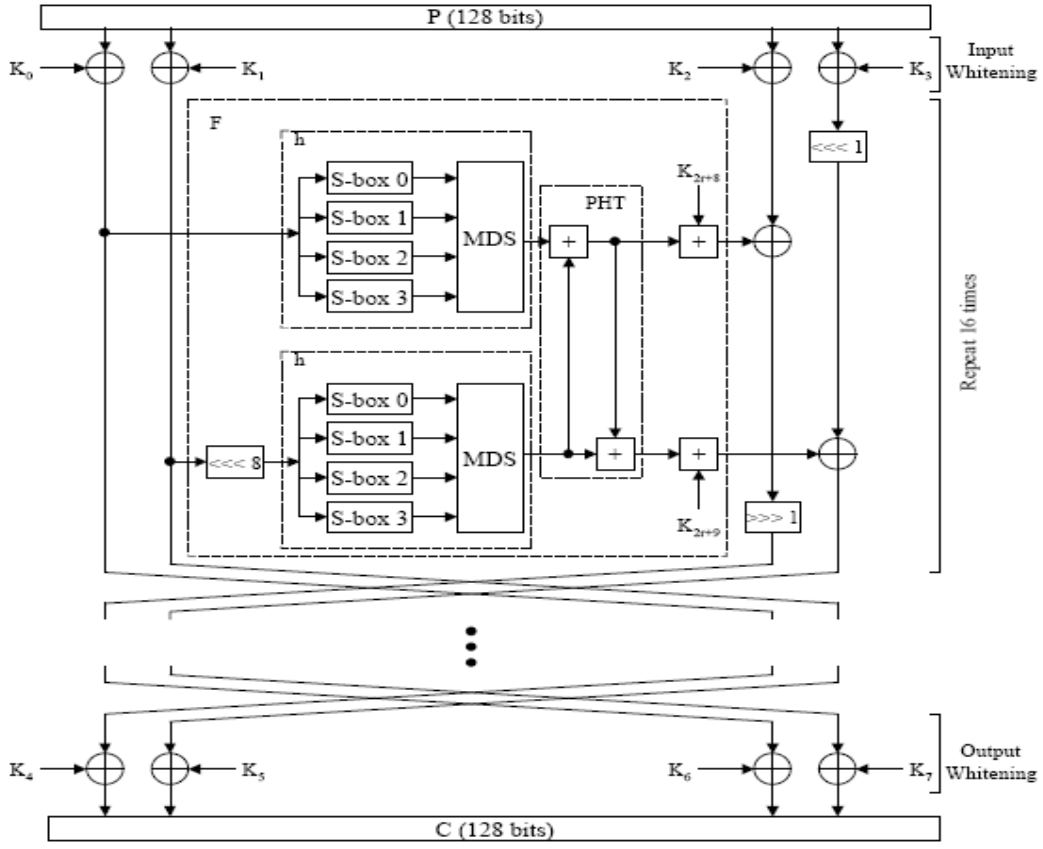
5. **Whitening**

Sebuah teknik meng-XOR-kan material kunci sebelum putaran pertama dan setelah putaran terakhir, digunakan oleh Markle dalam Khufu/Khafre, dan ditemukan oleh Revest untuk DES-X Whitening menambah tingkat kesulitan serangan pencarian kunci terhadap teks-kode, dengan menyembunyikan masukan spesifik terhadap putaran pertama dan putaran terakhir fungsi F.

Twofish meng-XOR-kan 128-bit upa-kunci sebelum putaran Feistel yang pertama, dan 128-bit lagi setelah putaran Feistel terakhir. Upa-kunci ini diperhitungkan dengan cara yang sama seperti upa-kunci putaran, tetapi tidak digunakan di tempat lain dalam kode.

6. **Penjadwalan Kunci:**

Penjadwalan kunci adalah proses pengubahan bit-bit kunci menjadi upa-kunci tiap putaran yang dapat digunakan oleh kode. *Twofish* memerlukan banyak material kunci dan memiliki penjadwalan kunci yang rumit. Untuk menfasilitasi analisis, penjadwalan kunci menggunakan primitif yang sama seperti fungsi putaran.



Gambar 2. 7.Blok diagram *Twofish*

1. Bit masukan sebanyak 128 bit akan dibagi menjadi empat bagian masing – masing 32 bit menggunakan konvensi little-Endian. Dua bagian bit akan menjadi bagian kanan, dan dua lainnya
2. Bit input akan di-XOR terlebih dahulu dengan empat bagian kunci, atau dengan kata lain mengalami proses whitening.

   $$R0,i = Pi \oplus Ki \quad i = 0,.....,3 \qquad (2.3)$$

   Dimana K adalah kunci, Ki berarti sub kunci yang ke-i.
3. Algoritma *Twofish* menggunakan struktur jaringan Feistel. Jaringan Feistel yang digunakan oleh *Twofish* terdiri atas 16 perulangan. Fungsi f pada algoritma *Twofish* terdiri atas beberapa tahap yaitu :
   a. Fungsi g, yang terdiri dari 4 s-box dan matriks MDS
   b. PHT (Pseudo-Hadamard Transformation) atau Transformasi Pseudo-Hadamard
   c. Penambahan hasil PHT dengan kunci

Pada setiap 16 putaran, dua kata pertama digunakan sebagai masukan kepada fungsi *F*, yang juga mengambil angka bulat itu sebagai masukan. Kata yang ketiga di-*XOR* dengan keluaran pertama F dan kemudian diputar ke kanan satu bit. Kata keempat diputar ke kiri satu bit kemudian di-*XOR* dengan kata keluaran *F* Yang kedua . Akhirnya, keduanya saling ditukar menghasilkan persamaan :

$$
\begin{aligned}
&(F_{r,0}\ F_{r,1}) = F(F_{r,0}F_{r,1}r)\\
&R_{r+1,0} = ROR(R_{r,2} \oplus F_{r,0,}1)\\
&R_{r+1,1} = ROL(R_{r,3,1}1) \oplus F_{r,1}\\
&R_{r+1,2} = R_{r,0}\\
&R_{r+1,3} = R_{r,1}
\end{aligned}
\qquad \text{.....(2.4)}
$$

untuk r = 0,…,15 di mana *ROR* dan *ROL* adalah berfungsi memutar argumentasi pertama (32-bit kata) ke kanan dengan angka bit-bit diindikasikan dengan argumentasi keduanya. Langkah whitening keluaran membatalkan `pertukaran' putaran terakhir dan meng *XOR* kata-kata dengan 4 kata dari kunci yang diperluas:

$$C_i = R_{16,(i+2) \bmod 4} \oplus K_{1+4}\ I = 0,.....,3 \text{ .....(2.5)}$$

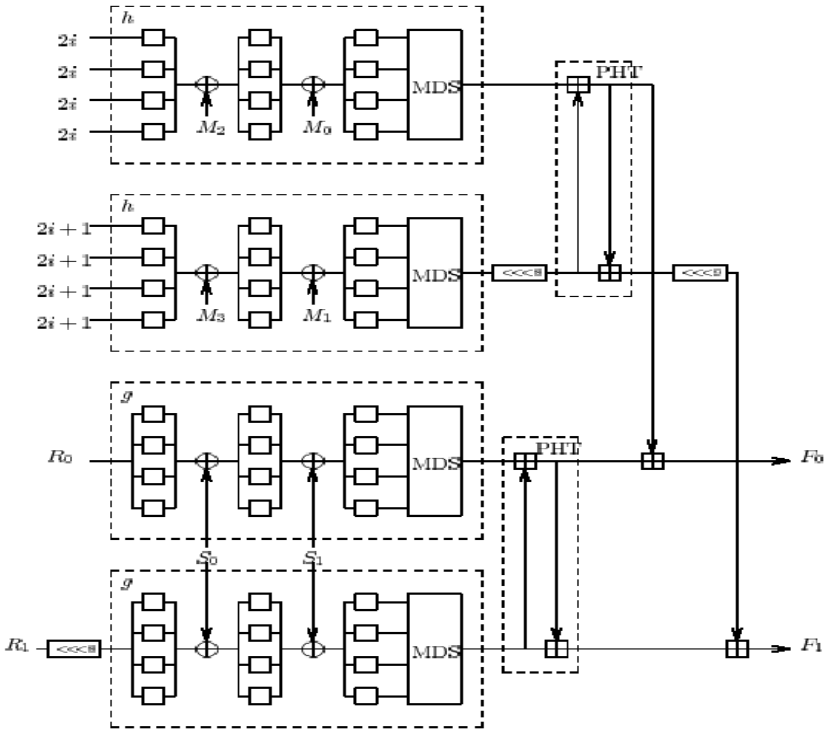
Empat kata dari ciphertext kemudian menulis seperti 16 *byte* $c_0,\ldots,c_{15}$ sama seperti menggunakan konversi little-endian untuk *plaintext*.

$$c_i = \left\lfloor \frac{C\lfloor i/4 \rfloor}{2^{8(i \bmod 4)}} \right\rfloor \bmod 2^8 \quad i = 0,.....,15 \text{ .....(2.6)}$$

Fungsi F adalah suatu permutasi *key-dependent* di atas nilai 64-bit. Untuk mengambil tiga argumentasi, dua kata masukan R_0 Dan R_1, dan angka bulat *r* digunakan untuk memilih *subkey* yang sesuai. R_0 yang dilewati fungsi g, menghasilkan T_0. *i* diputar 8 bit ke kanan kemudian melewati fungsi g untuk menghasilkan T_1. Hasil T_0 dan T_1 selanjutnya dikombinasikan dalam sebuah *PHT* danditambahkan 2 kata dari kunci yang diperluas menghasilkan persamaan :

$$
\begin{aligned}
&T_0 = g(R_0)\\
&T_1 = g(ROL(R_{1;}8))\\
&F_0 = (T_0 + T_1 + K_{2r+8}) \bmod 2^{32}\\
&F_1 = (T_0 + 2T_i + K_{2r+9}) \bmod 2^{32}
\end{aligned}
\qquad \text{.....(2.7)}
$$

Di mana (F_0, F_1) adalah hasil dari *F*. Kita juga menggambarkan fungsi *F'* untuk menganalisa. *F'* adalah identik dengan fungsi *F*, kalau tidak tambahkan beberapa blok kunci pada keluaran. (*PHT* masih dilakukan.)



Gambar 2. 8: Satu putaran fungsi *F* (kunci 128-bit)

Fungsi g membentuk jantungnya *Twofish*. Kata masukan *X* dipecah menjadi empat *byte*. Masing-masing *byte* dijalankan melewati *S-box key-dependent.* Masing-masing S-box adalah *bijective,* mengambil 8 bit masukan, dan menghasilkan 8 bit keluaran. Ke empat hasil diinterpretasikan sebagai vektor yang panjangnya 4 di atas *GF* (28), dan dikalikan dengan yang matriks *MDS* 4x4 (menggunakan bidang *GF* (28) untuk perhitungannya). Untuk menghasilkan vektor diinterpretasikan sebagai 32-bit kata sebaga adalah hasil dari g :

$$x_i = \left\lfloor X / 2^{8i} \right\rfloor \bmod 2^8 \; i = 0,....3 \quad \text{.....(2.6)}$$

$$y_i = s_i [x_i] i = 0,...3 \quad \text{.....(2.7)}$$

$$\begin{pmatrix} z_0 \\ z_1 \\ z_2 \\ z_3 \end{pmatrix} = \begin{pmatrix} . \\ MDS \\ . \end{pmatrix} . \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{pmatrix} \text{.....(2.8)}$$

$$Z = \sum_{i=0}^{3} z_i .2^8 \text{.....(2.9)}$$

Di mana s_i adalah *S-Box key-dependent* dan *Z* adalah hasil dari *g*. Untuk merumuskan dengan baik, kita harus menetapkan koresponden antara nilai-nilai *byte* dan elemen-elemen bidang *GF* (28). Kita merepresentasikan *GF* (2^8) sebagai *GF*(2)[x]/v(x) di mana $v(x)=x^8+x^6+x^5+x^3+1$ adalah suatu polynomial primitif dari 8 tingkat di atas GF(2). Unsur Bidang $a=\sum_{i=0}^{7} a_i x^i$ dengan a_i Є *GF* (2) adalah dikenal dengan nilai *byte* $\sum_{i=0}^{7} a_i x^i$. Ini adalah beberapa pengertian pemetaan "alamiah"; penambahan di dalam *GF* (28) berkorespondensi dengan suatu *XOR* dari *byte*s.

Matriks *MDS*-nya adalah sebagai berikut :

$$MDS = \begin{pmatrix} 01 & EF & 5B & 5B \\ 5B & EF & EF & 01 \\ EF & 5B & 01 & EF \\ EF & 01 & EF & 5B \end{pmatrix} \text{.....(2.10)}$$

Di mana elemen-elemen ditulis sebagai nilai-nilai *byte* hexadecimal.

Jadwal kunci harus menyediakan 40 kata dari kunci yang diperluas $K_0,\ldots,K_{39}$, dan 4 buah *S-Box key-dependent* yang digunakan di dalam fungsi g. *Twofish* didefinisikan untuk kunci-kunci dengan panjang N= 128, N= 192, dan N=

256. Beberapa kunci yang lebih pendek dari 256 bit dapat digunakan oleh lapisannya dari nol hingga yang lebih besar yang didefinisikan sebagai panjang kunci. Kita mendefinisikan k=N/64. Kunci M terdiri dari 8k *byte* $m_0,\ldots,m_{8k-1}$. *Byte-byte* adalah yang pertama diubah ke dalam 2k kata dari 32 bit masing-masing :

$$M_i = \sum_{j=0}^{3} m_{(4i+j)}.2^{8j} \; i = 0,\ldots 2k\text{-}1 \quad \text{.....(2.11)}$$

dan kemudian ke dalam dua vektor kata dari panjang k :

$$M_e = (M_0, M_2, \ldots, M_{2k-2})$$

$$M_o = (M_1, M_3, \ldots, M_{2k-2}) \quad \text{.....(2.12)}$$

Sepertiga vektor kata dari panjang *k* adalah juga diperoleh dari kunci itu.

Hal ini dilakukan dengan mengambil *byte-byte* kunci di dalam kelompok 8, menginterpretasikannya sebagai vektor di atas *GF* (28), dan mengalikannya dengan matrils 4x8 yang diperoleh dari suatu kode *R*. Masing-masing hasil dari 4 *byte* kemudian diinterpretasikan sebagai suatu kata 32 bit.. Kata-kata ini menyusun vektor yang ketiga :

$$\begin{pmatrix} s_{i,0} \\ s_{i,1} \\ s_{i,2} \\ s_{i,3} \end{pmatrix} = \begin{pmatrix} . \\ .RS \\ . \end{pmatrix} . \begin{pmatrix} m_{Si} \\ m_{Si+1} \\ m_{Si+2} \\ m_{Si+3} \\ m_{Si+4} \\ m_{Si+5} \\ m_{Si+6} \\ m_{Si+7} \end{pmatrix} \quad \text{.....(2.13)}$$

$$S_i = \sum_{j=0}^{3} S_{i,j}.2^{8j} \quad \text{.....(2.14)}$$

Keterangan

i =0,…,k-1

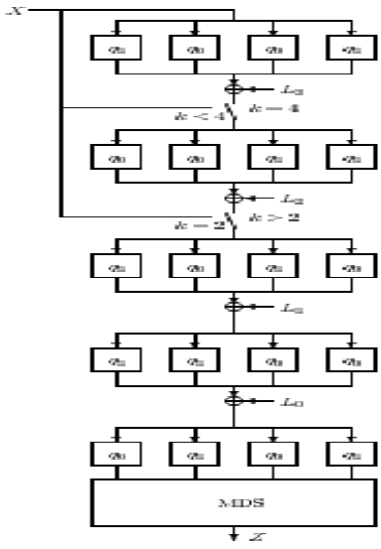
S =(S_k-1,S_{k-2},…,S_0)

Catat bahwa daftar S kata-kata itu di dalam order “terbalik". Karena Untuk perkalian matriks RS, *GF* (28) diwakili oleh *GF* (2)[x]/w(x), di mana w(x)= $x^8+x^6+x^3+x^2+1$ adalah polynomial primitif derajat dari 8 tingkat di atas GF(2). Pemetaan antara nilai-nilai *byte* dan elemen-elemen *GF* (28) menggunakan defnisi yang sama sebagaimana yang digunakan untuk perkalian matriks *MDS*. Dalam pemetaan ini, matriks RS ditunjukkan sebagai berikut :

$$RS = \begin{pmatrix} 01 & A4 & 55 & 87\,5A & 58 & DB & 9E \\ A4 & 56 & 82 & F31E & C6 & 68 & E5 \\ 02 & A1 & FC & C1\,47 & AE & 3D & 19 \\ A4 & 55 & 87 & 5A\,58 & DB & 9E & 03 \end{pmatrix} \quad \text{.....(2.15)}$$

Ke tiga vektor M_e, M_o, dan S ini membentuk basis dari jadwal kunci.

Twofish dapat menerima kunci-kunci dengan panjang *byte* di atas 256 bit.. Untuk ukuran-ukuran kunci yang tidak didefinisikan di atas, kunci diisi pada bagian akhir dengan nol *byte* kepada yang lebih panjang berikutnya. Sebagai contoh, suatu kunci 80-bit m_0,…,m_9 akan diperluas dengan mengatur m_i = 0 untuk i= 10,…,15 dan diperlakukan sebagaimana pada kunci 128-bit.



Gambar 2.9. Fungsi *h*

Fungsi h. Ini adalah suatu fungsi yang mengambil dua input-a 32-bit kata *X* dan sebuah daftar $L = (L_0,\ldots,L_{k-1})$ dari kata-kata *X* 32 dengan panjang k- dan menghasilkan satu kata pada outputnya. Ini adalah pekerjaan fungsi di dalam langkah-langkah k. Pada setiap langkah, empat *byte* itu masing-masing melintasi suatu *fixed S-Box*, dan di-*XOR* dengan sebuah *byte* yang diperoleh dari daftar. Akhirnya, *byte-byte* sekali lagi digeser melewati sebuah fixed S-box dan empat *byte* itu dikalikan dengan matriks *MDS* seperti halnya dalam g. Lebih formal kita memisah-misahkan kata-kata itu ke dalam *byte*s :

$$l_{i,j} = \lfloor L_i / 2^8 \rfloor \bmod 2^8$$
$$x_j = \lfloor X / 2^{8j} \rfloor \bmod 2^8 \quad \ldots\ldots(2.16)$$

untuk $i = 0,\ldots,k\text{-}1$ dan $j = 0,\ldots,3$. Kemudian urutan subtitusi dan *XOR* diterapkan :

$$y_{k,j} = x_j j = 0,\ldots,3 \quad \ldots\ldots(2.17)$$

jika $k = 4$ didapatkan:

$$\begin{aligned} y_{3,0} &= q_1\lfloor y_{4,0}\rfloor \oplus i_{3,0} \\ y_{3,1} &= q_0\lfloor y_{4,1}\rfloor \oplus i_{3,1} \\ y_{3,2} &= q_0\lfloor y_{4,2}\rfloor \oplus i_{3,2} \\ y_{3,3} &= q_1\lfloor y_{4,3}\rfloor \oplus i_{3,3} \end{aligned} \quad \text{.....(2.17)}$$

jika $k \geq 4$ didapatkan:

$$\begin{aligned} y_{2,0} &= q_1\lfloor y_{3,0}\rfloor \oplus i_{2,0} \\ y_{2,1} &= q_1\lfloor y_{3,1}\rfloor \oplus i_{2,1} \\ y_{2,2} &= q_0\lfloor y_{3,2}\rfloor \oplus i_{2,2} \\ y_{2,3} &= q_0\lfloor y_{3,3}\rfloor \oplus i_{2,3} \end{aligned} \quad \text{.....(2.18)}$$

Dalam seluruh kasus didapatkan :

$$\begin{aligned} y_0 &= q_1\left[q_0\left[q_0\lfloor y_{2,0}\rfloor \oplus i_{1,0}\right]\oplus i_{0,0}\right] \\ y_1 &= q_0\left[q_0\left[q_1\lfloor y_{2,1}\rfloor \oplus i_{1,1}\right]\oplus i_{0,1}\right] \\ y_2 &= q_0\left[q_1\left[q_0\lfloor y_{2,2}\rfloor \oplus i_{1,2}\right]\oplus i_{0,2}\right] \\ y_3 &= q_1\left[q_1\left[q_0\lfloor y_{2,3}\rfloor \oplus i_{1,3}\right]\oplus i_{0,3}\right] \end{aligned} \quad \text{.....(2.19)}$$

Di sini, q_0 dan q_1 ditetapkan permutasi di atas nilai 8-bit yang akan didefinisikan segera. Menghasilkan vektor yang merupakan perkalian matriks *MDS*, seperti halnya dalam fungsi g :

$$\begin{pmatrix} z_0 \\ z_1 \\ z_2 \\ z_3 \end{pmatrix} = \begin{pmatrix} . \\ .MDS \\ . \end{pmatrix} . \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{pmatrix} \quad \text{.....(2.20)}$$

$$\mathrm{Z} = \sum_{i=0}^{3} z_i . 2^{8i} \quad \text{.....(2.21)}$$

di mana Z adalah hasil dari h

Sekarang dapat didefinisikan *S-Box* dalam fungsi g oleh :

$g(x) = h(X; S)$

Hal itu adalah karena i= 0,…,3, *S-Box Key-Dependent* s_i dibentuk oleh pemetaan dari x_i ke y_i di dalam fungsi *h*, di mana daftar Lsama dengan vektor *S* yang diperoleh dari kunci itu.

Kata-kata dari kunci yang diperluas didefinisikan menggunakan fungsi *h* :

$$\rho = 2^{24} + 2^{16} + 2^8 + 2^0$$
$$A_i = h(2_{i\rho}, M_e)$$
$$B_i = ROL(h((2_i + 1)\,\rho, M_o), 8)$$
$$K_{2i} = (A_i + B_i) \bmod 2^{32}$$
$$K_{2i+1} = ROL((A_i + 2B_i) \bmod 2^{32}, 9) \quad \text{.....(2.22)}$$

Konstanta ρ digunakan untuk menduplikat *byte* yang mempunyai properti untuk i= 0,…,255, kata itu jika terdiri dari empat *byte* yang sama, masing-masing dengan nilai *i*. Fungsi h diberlakukan bagi kata-kata jenis ini. Untuk A_i nilai-nilai *byte* itu adalah 2*i*, dan argumentasi yang kedua dari h adalah M_e. B_i dihitung dengan cara yang sama menggunakan 2*i*+1 sebagai *byte* nilai dan M_o sebagai argumentasi yang kedua, dengan suatu putaran ekstra di atas 8 bit. Nilai-nilai A_i dan B_i Dua dikombinasikan dalam *PHT*. Salah satu dari hasil itu selanjutnya diputar dengan 9 bit. Kedua hasil tersebut membentuk dua kata kunci yang diperluas.

Permutasi q_0 dan q_1 adalah ditetapkan permutasi di atas nilai-nilai 8 bit. Mereka dibangun dari empat 4-bit permutasi yang masing-masing berbeda. Untuk nilai masukan x didefinisikan sebagai nilai output y sebagai berikut :

$$a_0, b_0 = \lfloor x/16 \rfloor, x \bmod 16$$
$$a_1 = a_0 \oplus b_0$$
$$b_1 = a_0 \oplus ROR_1(b_0, 1) \oplus 8a_0 \bmod 16$$
$$a_2, b_2 = t_0[a_1], t_1[b_1]$$

$a_3 = a_2 \oplus b_2$

$b_3 = a_2 \oplus ROR_1(b_2,1) \oplus 8a_2 \bmod 16$

$a_4, b_4 = t_2[a_3], t_3[b_3]$

$y = 16b_4 + a_4$ .....(2.23)

di mana ROR_4 adalah suatu fungsi yang serupa dengan *ROR* yang merupakan putaran nilai 4-bit. Pertama, *byte* dipecah menjadi dua bagian. Ini dikombinasikan dalam suatu bijective yang mencampur langkah. Masing-masing bagian kemudian melintasi *4-bitfixed S-Box*. Ini diikuti oleh yang lain. Akhirnya, dua bagian dikombinasikan kembali ke dalam satu *byte*. Untuk permutasi q_0 *S-Box* 4-bit :

t_0=[8 1 7 D 6 F 3 2 0 B 5 9 E C A 4]

t_1=[E C B 8 1 2 3 5 F 4 A 6 7 0 9 D]

t_2=[B A 5 E 6 D 9 0 C 8 F 3 2 4 7 1]

t_3=[D 7 F 4 1 2 6 E 9 B 3 0 8 5 C A]

di mana masing-masing *S-Box* 4-Bit diwakili oleh daftar masukan yang menggunakan notasi *hexadecimal*. (untuk masukan 0,…,15 didaftarkan dalam order.) Dengan cara yang sama, untuk q_1 *S-Box* 4-bit :

t_0=[2 8 B D F 7 6 E 3 1 9 4 0 A C 5]

t_1=[1 E 2 B 4 C 3 7 6 D A 5 F 9 0 8]

t_2=[4 C 7 5 1 6 9 A 0 E D 8 2 B 3 F]

t_3=[B 9 5 1 C 3 D E 6 4 7 F 2 0 8 A]

2.4 Kriptanalisis *Twofish*

Kriptoanalisis dapat pula diartikan sebagai seni atau ilmu untuk memecahkan cipherteks menjadi plainteks dengan memanfaatkan celah-celah keamanan sebuah sistem kriptografi. Hal inilah yang menjadikan kriptoanalisis dicap sebagai cara ilegal untuk menterjemahkan cipherteks. Berdasarkan kegiatan pihak penyerang, serangan yang terjadi dapat dibedakan menjadi dua jenis yaitu:

a) **Serangan pasif**, adalah serangan dimana penyerang hanya memonitor saluran komunikasi. Penyerang pasif hanya mengancam kerahasiaan data.

b) **Serangan aktif**, adalah serangan dimana penyerang mencoba untuk menghapus, menambahkan, atau dengan cara yang lain mengubah transmisi pada jalur komunikasi. Penyerang aktif akan mengancam integritas data dan otentikasi, juga kerahasiaan.

Sudah di atas seribu jam waktu yang digunakan untuk mencoba ke kriptanalisis *Twofish*. Suatu paparan tentang serangan yang berhasil adalah sebagai berikut :

- *Twofish 5-round* (tanpa *post-whitening*) dengan $2^{22,5}$ pilihan pasangan *plaintext* dan 2^{51} perhitungan fungsi g.
- *Twofish 10-round* (tanpa pre dan *post whitening*) dengan sebuah serangan pilihan kunci, menuntut 2^{32} pilihan *plaintext*s dan sekitar 2^{11} pilihan *plaintext* adaptip, serta sekitar 2^{32} pekerjaan.

Kenyataan bahwa *Twofish* nampak tahan terhadap serangan *related-key* dan memiliki hasil yang paling menarik. Pemecahan kriptanalisis secara konvensional mengijinkan suatu penyerang untuk mengendalikan masukan *plaintext* dan ciphertext ke dalam cipher. Kriptanalisis *Related-key* memberi penyerang itu suatu cara tambahan dalam sebuah cipher: jadwal kunci. Sebuah *cipher* adalah tahan terhadap serangan dengan kunci terkait dengan teknik lebih sederhana yang hanya melibatkan *plaintext* itu dan ciphertext. Berdasarkan 19 kali penelitian yang pernah dilakukan, diduga bahwa tidak ada lagi serangan yang efisien atas *Twofish* selain kekuatan fisik. Diperkirakan bahwa serangan yang

paling efisien terhadap *Twofish* dengan suatu kunci 128-bit mempunyai kompleksitas 2^{128}, sedangkan serangan yang paling efisien terhadap *Twofish* dengan sebuah kunci 192-bit mempunyai kompleksitas 2^{192}, dan serangan yang paling efisien terhadap *Twofish* dengan sebuah kunci 256-bit mempunyai kompleksitas 2^{256}.

Kekuatan algoritma kriptografi moderen terletak pada kuncinya yaitu berupa deretan karekter atau bilangan bulat yang dijaga kerahasiaannya, kunci ini dianalogikan dengan penggunaan PIN (*Personal Identity Number*) pada ATM. Sementara itu *Twofish* merupakan algoritma kuat yang sampai saat ini dinyatakan aman karena masih belum ada serangan kriptanalis yang benar-benar dapat mematahkan algoritma ini. *Twofish* memiliki resistensi yang tinggi terhadap *related-key attack* dan dapat ditembus henya dengan menggunakan *brute force*

Untuk menebus sebuah algoritma kriptografi dengan panjang kunci 64-bit saja dibutuhkan waktu sekitar 2^{64} atau dengan kata lain 72.057.594.037.927.936 kali kemungkinan kunci Jika diandaikan dalam satu detik dapat dicobakan satu juta kemungkinan kunci, maka akan diperlukan waktu sekitar 2284 tahun untuk menemukan kunci yang benar, nah bisa dibayangkan jika algoritma *Twofish* dengan panjang kunci diatas 128 bit akan memerlukan waktu yang lebih lama lagi untuk memecahkan kuncinya. (Ariyus, 2008).

Pada akhirnya ada 3 aspek atau kondisi yang menyebabkan algoritma kriptografi dianggap aman:

1. Apabila biaya yang dibutuhkan untuk menembus kunci lebih besar bila dibandingkan dengan informasi yang akan didapat.

2. Apabila waktu untuk menembus kunci lebih lama bila dibandingkan dengan validasi informasi yang akan ditembus.
3. Apabila *chipertext* yang dihasilkan oleh suatu algoritma kriptografi lebih sedikit dari *chipertext* yang diperlukan untuk menembus algoritma tersebut.

2.5 Bilangan Biner

Sistem bilangan biner adalah susunan bilangan yang mempunyai basis 2 sebab sistem bilangan ini menggunakan dua nilai koefisien yang mungkin yaitu **0** dan **1**. berikut erupakan tabel yang menggambarkan basis bilangan biner dan bilangan desimal.

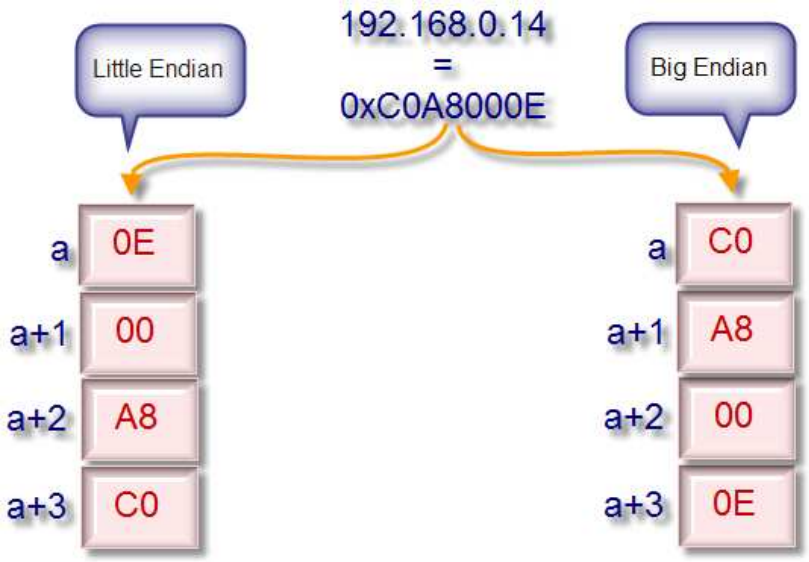
Tabel 2.1 Basis Bilangan Desimal dan Biner

Bilangan Desimal (base 10)	**Bilangan Biner (base 2)**
0	**0000**
1	**0001**
2	**0010**
3	**0011**
4	**0100**
5	**0101**
6	**0110**
7	**0111**
8	**1000**
9	**1001**
10	**1010**
11	**1011**
12	**1100**
13	**1101**
14	**1110**
15	**1111**

2.6 Little Endian

Secara umum terdapat dua teknik pembagian bit-bit binary antara lain big-endian dan little-endian. Terdapat perbedaan antara keduanya antara lain adalah: Pada notasi little endian, *byte* rendah ditulis duluan, baru kemudian *byte* tinggi.

Sedangkan pada notasi big endian, *byte* tinggi ditulis duluan, baru kemudian *byte* tinggi. Perhatikan gambar di bawah ini untuk melihat perbedaan little dan big endian.



Gambar 2.9 Perbedaan antara *Little-endian* dan *Big-endian*

Pada gambar di atas IP address 192.168.0.14 dalam hexa adalah C0.A8.00.0E disimpan dalam notasi little endian dan big endian. *Byte* paling rendah, yaitu 0×0E disimpan di alamat memori paling rendah pada sistem little endian. Sedangkan pada big endian, *byte* paling tinggi 0xC0 disimpan di alamat memori paling rendah.

2.7 Citra Digital

Salah satu daya tarik manusia dalam menikmati suatu objek adalah adanya unsur gambar *(image)*. Gambar digital merupakan dokumen berbentuk *file* yang dihasilkan melalui perangkat elektronik atau media digital. Gambar digital berupa sekumpulan titik yang disusun dalam bentuk matriks, dan nilainya menyatakan suatu derajat kecerahan (derajat keabuan/*gray-scale*). Derajat keabuan 8 bit

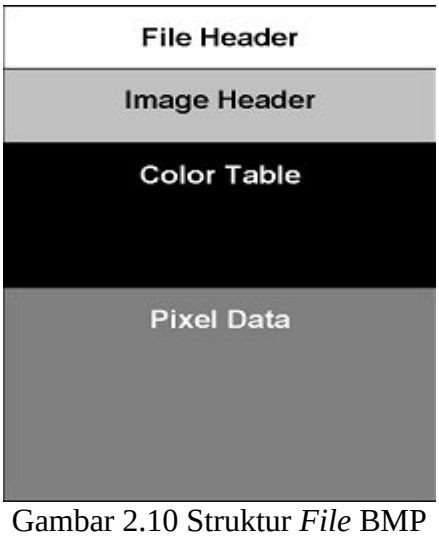
menyatakan 256 derajat kecerahan. Pada gambar berwarna nilai setiap titiknya adalah nilai derajat keabuan pada setiap kompoen warna RGB. Bila masing-masing komponen R,G dan B mempunyai 8 bit, maka satu titik dinyatakan dengan (8+8+8)=24 bit atau 2^{24} derajat keabuan.

Secara garis besar, gambar dapat dibagi menjadi menjadi beberapa tipe atau format, diataranya:

1. BMP *(BitMap Graphics)*.

   Tipe file BMP umum digunakan pada sistem operasi *Windows*. Kelebihan file BMP adalah dapat dibuka oleh hampir semua program pengolah gambar. Baik file BMP yang terkompresi maupun tidak terkompresi, file BMP memiliki ukuran yang jauh lebih besar dari pada tipe-tipe yang lain. Struktur file BMP terdiri dari 4 bagian, yaitu: *File Header, Image Header, Color Table* dan *Data Pixel*. Header file BMP (*File Header* + *Image Header* + *Color Table*) biasanya sebesar 54 byte.

   Struktur file BMP dapat dilihat dari gambar berikut:



Gambar 2.10 Struktur *File* BMP

2. JPG / JPEG (*Joint Photographic Expert Group*)

Format ini didesain untuk gambar-gambar dengan keadalaman warna 24-bit. *File* JPG menggunakan teknik kompresi yang menyebabkan kualitas gambar turun (*lossy compression*). Setiap kali menyimpan ke tipe JPG dari tipe lain, ukuran gambar biasanya mengecil, dan kualitasnya turun dan tidak dapat dikembalikan lagi. Ukuran *file* BMP dapat turun menjadi sepersepuluh setelah dikonversi menjadi JPG. Meskipun dengan penurunan kualitas gambar, pada gambar-gambar tertentu (misalnya pemandangan), penurunan kualitas gambar hampir tidak terlihat mata.

3. GIF (*Graphics Interchange Format*)

File GIF memungkinkan penambahan warna transparan dan dapat digunakan untuk membuat animasi sederhana, tetapi saat ini standar GIF hanya maksimal 256 warna saja. *File* ini menggunakan kompresi yang tidak menghilangkan data *(lossles compression)* tetapi penurunan jumlah warna menjadi 256 sering membuat gambar yang kaya warna seperti pemandangan menjadi tidak realistis.

4. PNG (Portable Network Graphics).

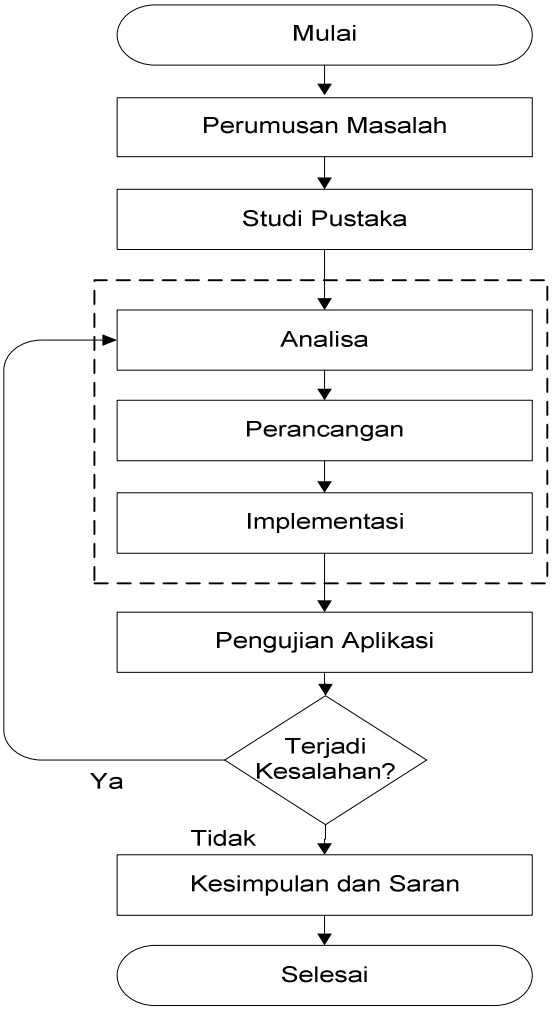
Tipe file PNG merupakan solusi kompresi yang *powerfull* dengan warna yang lebih banyak (24 bit RGB + *alpha*). Berbeda dengan JPG yang menggunakan teknik kompresi yang menghilangkan data, *file* PNG menggunakan kompresi yang tidak menghilangkan data (*lossles compression*). Kelebihan *file* PNG adalah adanya warna transparan dan alpha. Warna alpha memungkinkan sebuah gambar transparan, tetapi

gambar tersebut masih dapat dilihat mata seperti samar-samar atau bening. File PNG dapat diatur jumlah warnanya 64 bit (*true color +alpha*) sampai *indexed color* 1 bit. Dengan jumlah warna yang sama, kompresi file PNG lebih baik daripada GIF, tetapi memiliki ukuran *file* yang lebih besar daripada JPG.

BAB III

METODOLOGI PENELITIAN

Metodologi penelitian menguraikan tahapan seluruh kegiatan yang dilaksanakan selama kegiatan penelitian berlangsung. Adapun langkah-langkah yang dilalui dalam pelaksanaan penelitian ini adalah sebagai berikut:



Gambar 3.1 Diagram Alir Metodologi Penelitian

3.1 Perumusan Masalah

Dengan memanfaatkan informasi-informasi yang didapat dari penelitian pendahuluan yang telah dilakukan, maka dilakukan tahap berikutnya yaitu mengidentifikasi masalah. Pada tugas akhir ini masalah yang akan diidentifikasi adalah melakukan studi analisis algoritma *twofish* untuk diimplementasikan dalam penyandian citra digital bertipe bitmap 24 bit.

3.2 Studi Pustaka

Tahap ini merupakan langkah yang dilakukan untuk mendapatkan teori-teori lanjutan yang dibutuhkan, yaitu dengan melakukan Studi Pustaka (*Library Research*). Studi pustaka ini bertujuan untuk mendapatkan dasar-dasar pengetahuan yang akan diterapkan dalam penelitian dan memperoleh informasi dalam tahap persiapan penelitian ini, maka dipelajari bahan pustaka yang ada kaitannya dengan penelitian algoritma *twofish*, penyandian citra dan pemrograman menggunakan *Borland Delphi 7*.

3.3 Analisa

Tahap ini merupakan tahap analisa terhadap algoritma yang digunakan berdasarkan teori-teori yang berkaitan. Tahap ini meliputi analisa terhadap blok diagram *twofish*.

Analisa ini akan berguna untuk menggambarkan secara keseluruhan mengenai konsep dasar algoritma *twofish.* Sehingga dapat memudahkan penelitian dalam mengimplementasikannya.

3.4 Perancangan Program

Pada tahap ini, dilakukan perancangan terhadap perangkat lunak yang akan dibangun. Perancangan ini meliputi perancangan struktur menu dan perancangan *interface*.

3.5 Implementasi

Tahap implementasi merupakan tahap penerjemahan hasil analisa ke dalam bentuk *coding* sesuai dengan hasil perancangan yang telah dibuat. Bahasa pemograman yang digunakan untuk membangun perangkat lunak algoritma yaitu *Borland Delphi 7*.

3.6 Pengujian

Selanjutnya dilakukan pengujian terhadap aplikasi yang telah dibangun agar dapat diketahui hasilnya. Jika terdapat *error*, atau proses enkripsi maupun dekripsi tidak berhasil, maka proses akan kembali ke tahap analisis untuk dilakukan analisa ulang.

3.7 Kesimpulan dan saran

Berdasarkan hasil pengujian dihasilkan kesimpulan yang sesuai dengan rumusan masalah dan tujuan yang akan dicapai, serta saran-saran yang diperlukan untuk pengembangan selanjutnya

BAB IV

ANALISA DAN PERANCANGAN

4.1 Analisa

Analisis perangkat lunak dalam membangun aplikasi kriptografi untuk penyandian citra menggunakan algoritma *Twofish* meliputi analisis masalah, analisis metode, analisis kebutuhan sistem dari aplikasi yang akan dibuat, sehingga aplikasi yang dibangun sesuai dengan maksud dan tujuan yang ingin dicapai dalam penelitian ini.

4.1.1 Analisa Masalah

Kriptografi adalah ilmu yang mempelajari teknik-teknik matematis yang berhubungan dengan aspek keamanan informasi seperti : keabsahan, integritas data, serta autentifikasi data.

Pada masa yang akan datang tidak menutup kemungkinan data yang disimpan didalam memori penyimpanan dapat berbentuk *file* gambar. Hal ini disebabkan karena gambar akan lebih mudah untuk menjelaskan sesuatu dari pada data yang bebentuk teks. Untuk itu perlu dibangun suatu aplikasi yang dapat mengamankan data dalam bentuk *file* gambar. Maka timbullah ide untuk merancang aplikasi kriptografi pada citra menggunakan algoritma *twofish*.

4.1.2 Analisa Kebutuhan Sistem

Setelah melakukan analisa, maka dapat diketahui kebutuhan sistem yang akan dibangun seperti kebutuhan masukan, kebutuhan fungsi dan proses, keluaran proses yang akan dibuat, sehingga aplikasi yang dibangun sesuai dengan yang diharapkan.

4.1.2.1 Kebutuhan Masukkan

Data masukkan yang dibutuhkan dalam membangun aplikasi ini berupa:

1. *File* citra dengan format bmp
2. Kata kunci atau yang lebih di kenal dengan *Password* yang digunakan untuk pengamanan dalam proses enkripsi
3. Jumlah proses enkripsi yang merupakan pengamanan dalam proses enkripsi selain *password.*

4.1.2.2 Kebutuhan Fungsi atau Proses

1. Proses pemilihan *file.*

   Merupakan proses pemilihan *file* dimana *file* yang dapat dienkripsi berupa *file* dengan format bmp.

2. Proses pemilihan kunci (*password).*

   Proses ini merupakan fasilitas pengamanan dari kriptografi dimana pengguna atau *user* harus meng*input*kan kata kunci yang juga digunakan untuk proses dekripsi.

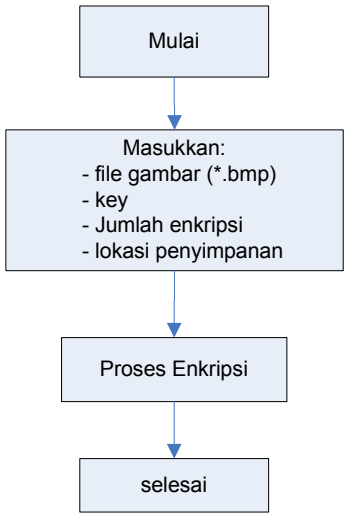
3. Proses pemilihan jumlah Enkripsi

   Sama halnya seperti pemilihan kunci diatas hanya saja pada proses kali ini pengguna atau *user* harus meng*input*kan berapa jumlah enkripsi.

Jumlah enkripsi harus sesuai dengan jumlah dekripsi, jika terdapat perbeaan maka gambar yang semula di enkripsi tidak akan kembali seperti gambar semula, namun aplikasi akan tetap menjalakan perintah.

4. Proses enkripsi

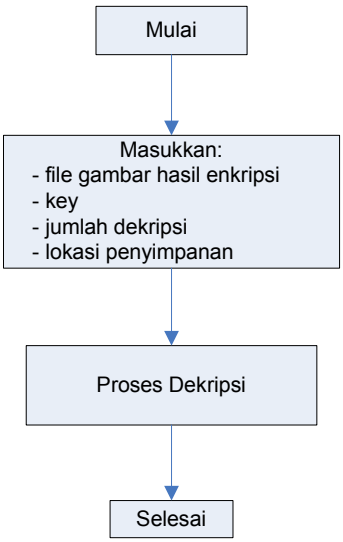
   Proses enkripsi berisi tombol enkripsi, setelah pemilihan *file* dilanjutkan dengan memasukkan kata kunci dan pemilihan jumlah enkripsi maka proses enkripsi dapat dilanjutkan dengan syarat *file* bertipe bmp. Berikut merupakan flowchart penyandian citra dengan algoritma *Twofish*:



Gambar 4.1 *Flowchart* Proses Enkripsi

5. Proses Dekripsi

   Proses dekripsi hampir sama dengan proses enkripsi. Dimana *file* yang didekrip merupakan *file* yang telah dienkrip sebelumnya dengan kunci dan jumlah enkripsi yang sama. Bila terdapat perbedaan antara kunci proses enkripsi dengan dekripsi maka proses tetap dapat dijalankan tetapi tidak menemukan hasil yang diinginkan.



Gambar 4.2 *Flowchart* proses dekripsi

4.1.2.3 Analisa Keluaran Proses

Setelah proses kriptografi selesai dilakukan, aplikasi ini akan menampilkan sebuah keluaran yang sama seperti *file* aslinya hanya saja gambar yang dihasilkan telah berubah menjadi gambar yang sudah tidak beraturan dan tidak dikenali lagi, karena pixel-pixel dari gambar tersebut telah dienkripsi. Untuk membalikkan gambar seperti keadaan semula harus dilakukkan proses dekripsi.

4.1.3 Analisa Metode

Karakteristik sebuah *file* didalam memori penyimpanan adalah memiliki *header file,header file* ini adalah *extention* pada saat pemanggilan *file.* Pada kasus aplikasi yang dirancang *header file* tidak ikut dienkripsi sehingga setelah proses enkripsi selesai gambar masih dapat dibuka karena *header file*-nya tidak berubah.

Untuk kasus penyandian citra menggunakan algoritma *twofish* langkah-langkah setiap bloknya dapat dilihat sebagai berikut:

Step 0 : Masukkan citra yang akan disandikan (*.bmp)

Step 1 : Pecah nilai RBG pixel menjadi bentuk biner

Step 2 : Blok-blok biner menjadi 128 bit

Step 3 : Pecah blok menjadi 4 bagian (A,B,C dan D) masing-masing bernilai 32 bit, kemudian diXORkan dengan kunci *(input whitening)*.

Step 4 : A dan B masuk ke struktur jaringan Feistel, sebelumnya B digeser ke kiri sejauh 8 bit $\boxed{<<< 8}$,fungsi F pada algoritma *twofish* terdiri dari beberapa tahap yaitu :

a. Fungsi g yang terdiri dari 4 S-box dan matrik MDS
b. Transformasi Pseudo Hadamard (PHT)
c. Penambahan hasil PHT dengan kunci

Step 5 : A diXORkan dengan C kemudian digeser kekiri sejauh 1 bit $\boxed{<<<1}$ Menghasilkan C', B diXORkan dengan D yang telah digeser kekiri sejauh 1 bit $\boxed{<<<1}$ sebelumnya menghasilkan D'.

Step 6 : A digeser ke C dan B digeser ke D, sedangkan C' digeser ke A dan D' digeser ke B

Step 7 : Lakukan Step 4 – 6 sebanyak 15 kali perulangan

Step 8 : Hasil dari Step 7 diXORkan dengan kunci *(ouput whitening)*

Step 9 : Hasil step 8 merupakan cipertext 1 blok algoritma *twofish*.

Langkah-langkah 1 putaran algoritma *twofish* adalah sebagai berikut:

Berikut ini merupakan contoh dari proses enkripsi algoritma *twofish*.pada citra bertipe *.bmp dengan nilai RGB / plaintext yang diperoleh untuk blok pertama sebagai berikut **119 131 129 116 128 126 98 110 108 86 98 96 61 73 71 76** yang merupakan satu putaran proses enkripsi. RGB tersebut dikelompokkan menjadi 4 bagian masing bernilai 32 bit. **119 131 129 116** | **128 126 98 110** | **108 86 98 96** | **61 73 71 76** dalam bentuk biner angka RGB tersebut menjadi **01110101 10000011 10000001 01110100** | **10000000 01111110 01100010 01101110** | **01101100 01010110 01100010 01100000** | **00111101 01001001 01000111 01001100** biner tersebut akan mengalami proses whitening atau di XOR-kan dengan empat bagian kunci. Misalkan kunci yang digunakan dalam biner sebagai berikut : **10010101 01110101 01010111 01010111.**

Plaintext a = **01110101 10000011 10000001 01110100** di- XOR-kan dengan kunci K **10010101 01110101 01010111 01010111**

= **11100000 11110110 11010110 00100011**

Plaintext b = **10000000 01111110 01100010 01101110** di- XOR-kan dengan kunci K **10010101 01110101 01010111 01010111**

= **00010101 00001011 00110101 00111001**

Plaintext b digeser sejauh 8 bit ke kiri <<< 8 maka hasilnya **00001011 00110101 00111001 00000000** plaintext a dan plaintext b masuk ke jaringan feistel dimana *twofish* memiliki jaringan feistel dengan 16 kali perulangan, fungsi F pada algoritma *twofish* terdiri dari beberapa tahap:

a. S-Box plaintext a dibagi menjadi 4 bagian s-box yaitu

- **11100000 = 224_{10}** didefenisikan sebagai S-box 0

- **11110110 = 246_{10}** didefenisikan sebagai S-box 1

- **11010110 = 214_{10}** didefenisikan sebagai S-box 2

- **00100011 = 35_{10}** didefenisikan sebagai S-box 3

Kemudian masing-masing S-box dikalikan dengan Matrix Distance Separable (MDS) lihat persamaan (X) pada bab sebelumbya, dimana S-box 0 didefenisikan sebagai y_0, S-box 1 didefenisikan sebagai y_1, S-box 2 didefenisikan sebagai y_2 dan S-box 3 didefenisikan sebagai y_3. Hasil perkalian antara S-box dan matrix MDS di defenisikan sebagai z_0, z_1, z_2 dan z_3. seperti pada persamaan (VIII) pada bab sebelumnya.

Hasil perkalian antara kedua matriks adalah :

Z_0 = **0100 0000**

Z_1 = **1011 1100**

Z_2 = **0110 1010**

Z_3 = **1110 1110**

b. S-box plaintext b dibagi menjadi 4 bagian S-box yaitu:

- **00001011 = 11_{10}** didefenisikan sebagai S-box 0

- **00110101 = 53_{10}** didefenisikan sebagai S-box 1

- **00111001 = 57_{10}** didefenisikan sebagai S-box 2

- **00000000 = 0_{10}** didefenisikan sebagai S-box 3

Dikalikan dengan matrik MDS maka hasilnya :

Z_0 = **0010 0010**

Z_1 = **0000 0010**

Z_2 = **1110 1010**

Z_3 = **0000 0000**

Untuk proses Transformation Pseudo-Hadamard (PHT) lihat persamaan (II) pada bab sebelumnya.

a' = **0100 0000 1011 1100 0110 1010 1110 1110 + 0010 0010 0000 0010 1110 1010 0000 0000** mod 2^{32}

= **11100000 00101011 10101001 00011100**

b' = **0100 0000 1011 1100 0110 1010 1110 1110 + 2(0010 0010 0000 0010 1110 1010 0000 0000)** mod 2^{32}

= **10101001 10001011 11101111 01010000**

a' diXORkan dengan kunci

= **11100000 00101011 10101001 00011100** XOR **10010101 01110101 01010111 01010111**

= **01110101 01011110 11111110 01001011**

b' diXORkan dengan kunci

= **10101001 10001011 11101111 01010000** XOR **10010101 01110101 01010111 01010111**

= **00111100 11111110 10111000 00000111**

C diXORkan dengan kunci

C = **01101100 01010110 01100010 01100000** diXORkan dengan **10010101 01110101 01010111 01010111**

= **11111001 00100011 00110101 00110111**

a' diXORkan denga C

= **01110101 01011110 11111110 01001011** diXORkan dengan **11111001 00100011 00110101 00110111**

= **10001100 01111101 11001011 01111100**

Hasil dari keduanya digeser kekiri sejauh 1 bit <<<1

C' = **00011000 11111011 10010110 11111000**

D diXORkan dengan kunci

= **00111101 01001001 01000111 01001100** diXORkan dengan **10010101 01110101 01010111 01010111**

= **10101000 00111100 00010000 00011011**

D' digeser ke kiri sejauh 1 bit <<<1

D' = **0101000 00111100 00010000 000110110**

D' diXORkan dengan b'

= **0101000 00111100 00010000 000110110** diXORkan dengan **00111100 11111110 10111000 00000111**

= **01101100 10000110 10011000 00110001**

A dan B ditukar dengan C' dan D'. Hal ini dilakukan sebanyak 16 kali perulangan, setelah selesai maka diXORkan kembali dengan kunci sebagai Output *whitening*. (misalkan A,B,C' dan D' telah dilakukan perulangan sebanyak 16 kali)

A = **01110101 01011110 11111110 01001011** diXORkan dengan **10010101 01110101 01010111 01010111**

= **11100000 00101011 10101001 00011100**

= **224 43 169 28**

B = **00111100 11111110 10111000 00000111** diXORkan dengan **10010101 01110101 01010111 01010111**

= **10101001 10001011 11101111 01010000**

= **169 139 240 80**

C’ = **00011000 11111011 10010110 11111000** diXORkan dengan **10010101 01110101 01010111 01010111**

= **10001101 10001110 11000001 10101111**

= **141 142 193 175**

D’ = **00001101 00001001 10010101 11110111** diXORkan dengan **10010101 01110101 01010111 01010111**

= **01101100 10000110 10011000 00110001**

= **108 134 152 49**

Maka cipertext 1 blok menjadi **224 43 169 28 169 139 240 80 141 142 193 175 108 134 152 49**

4.2 Perancangan

Perancangan perangkat lunak dalam membangun aplikasi kriptografi untuk penyandian citra menggunakan algoritma *twofish* meliputi perancangan antar muka proses enkripsi dan perancangan antarmuka proses dekripsi.

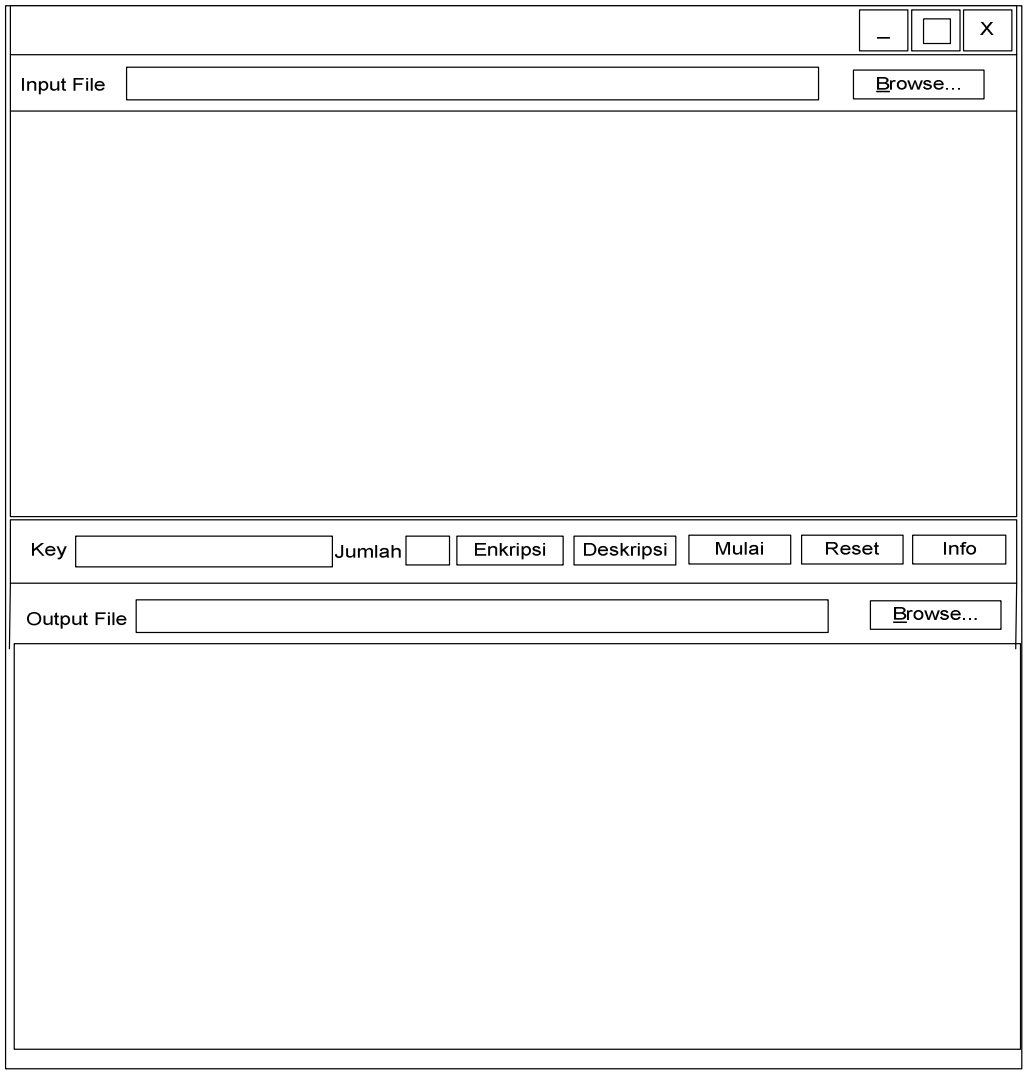
4.2.1 Perancangan Antarmuka Aplikasi Yang Akan Dibangun

Antarmuka atau *interface* merupakan suatu sarana yang memungkinkan terjadinya interaksi antara manusia dan komputer. Oleh sebab itu, *interface* dari sebuah perangkat lunak yang akan dibangun harus bersifat *user friendly* yang

bertujuan agar pengguna (*user*) dapat mengerti dengan mudah dan memahami cara menggunakan perangkat lunak ini.

4.2.2 Perancangan Antarmuka Aplikasi

Gambaran antarmuka (*interface*) yang dibutuhkan aplikasi yang akan dibangun ini terdiri dari beberapa bagian yang diakses *user*. Salah satu perancangan tampilan enkripsi pada kriptografi ini adalah sebagai berikut pada gambar 4.3 :



Gambar 4.3 Rancangan Tampilan

Tabel 4.1 Tabel Keterangan Tampilan

Objek	**Properti**	**Pengaturan**
Label 1	Caption	Input *file*1
Label 2	Caption	Key
Label 3	Caption	Jumlah
Label 4	Caption	Output *File*
Label 5	Caption	Menampilkan gambar awal
Label 6	Caption	Menampilkan gambar setelah di enkripsi
button 1	Caption	Browse1
button 2	Caption	Enkripsi
button 3	Caption	Dekripsi
button 4	Caption	Info
button 4	Caption	Browse2
Textbox 1	Font	MS Sans Serif
Textbox 2	Font	MS Sans Serif
Textbox 3	Font	MS Sans Serif
Textbox 4	Font	MS Sans Serif

BAB V

IMPLEMENTASI DAN PENGUJIAN

5.1 Implementasi Sistem

Implementasi merupakan tahapan dimana sistem sudah bisa dioperasikan. Hal ini dilakukan setelah penulisan kode program. Pada tahap implementasi sistem ini, diharapkan sistem yang telah dirancang siap untuk dioperasikan pada keadaan yang sebenarnya, sehingga akan diketahui apakah sistem yang dibuat benar-benar sesuai seperti yang diharapkan. Implementasi perangkat lunak ini dibuat dan dibangun dengan bantuan bahasa pemrograman *Borland Delphi 7.0*.

5.1.1 Alasan Pemilihan Perangkat Lunak

Pemilihan perangkat lunak ini didasarkan pertimbangan sebagai berikut:

1. *borland delphi 7.0* merupakan *event-driven programming* (pemograman terkendali kejadian) artinya program menunggu sampai adanya respon dari pemakai berupa *event* atau kejadian tertentu (tombol diklik, menu dipilih, dan lain-lain). Ketika *event* terdeteksi, kode yang berhubungan dengan *event* (prosedur *event*) akan dijalankan.
2. *borland delphi 7.0* mempunyai kemampuan untuk memanfaatkan *windows* dan tidak memerlukan pemrograman khusus untuk menampilkan jendela (*window*), dan cara penggunaannya berbasis *visual* seperti aplikasi *windows* lainnya.

3. *borland delphi 7.0* dapat digunakan untuk menguji program (*debugging*) dan menghasilkan program akhir EXE, yang bersifat *executable* atau dapat langsung dijalankan.
4. *borland delhpi 7.0* sering dieksploitasi untuk menunjang efesiensi bisnis karena kemudahan mendesain *user interface*.

5.1.2 Alasan Pemilihan *File* BMP

Pada dasarnya teknologi algoritma modern seperti *twofish* dan algoritma kriptografi modern lainnya yang dirancang mampu mengenkripsi semua *file*, namun pada tugas akhir ini penulis hanya membatasi *file* bmp sebagai data *input*-an karena *file* bmp merupakan *file* yang banyak digunakan dan hampir dapat dibuka disemua program pengolah gambar, *file* bmp juga mudah untuk disandikan karena ukuran *byte*-nya dapat dihitung tinggi dan lebarnya.

Pada aplikasi yang dirancang *header file* tidak ikut dienkripsi, sehingga pada saat *file* telah selesai dienkripsi gambar masih tetap dapat dibuka. Algoritma *twofish* mengambil piksel dari *file* bmp sebagai masukan (*plainteks*) untuk dienkripsi dan mengubahnya menjadi piksel yang acak sehingga gambar yang telah dienkripsi sudah tidak dapat dikenali lagi.

5.1.3 Batasan Implementasi

Batasan implementasi pada penulisan tugas akhir ini adalah

1. Aplikasi ini tidak menyimpan kunci proses enkripsi, sehingga apabila *user* lupa atau kehilangan kunci maka tidak dapat membalikkan gambar ke gambar aslinya.

2. Tidak terdapat pesan error apabila *user* salah memasukkan kunci, aplikasi akan tetap menjalankan proses. Tapi hasil yang dikeluarkan tidak sama seperti gambar aslinya.

5.1.3 Lingkungan Implementasi

Implementasi dilakukan pada lingkungan perangkat keras dan lingkungan perangkat lunak.

1. Perangkat Keras

   Perangkat keras yang digunakan mempunyai spesifikasi sebagai berikut:

   a. *Processor* Intel Core Duo 1,83 GHz
   b. *Memory* 512 MB
   c. *Harddisk* berkapasitas 80 GB
   d. Monitor
   e. *Keyboard*
   f. *Mouse*

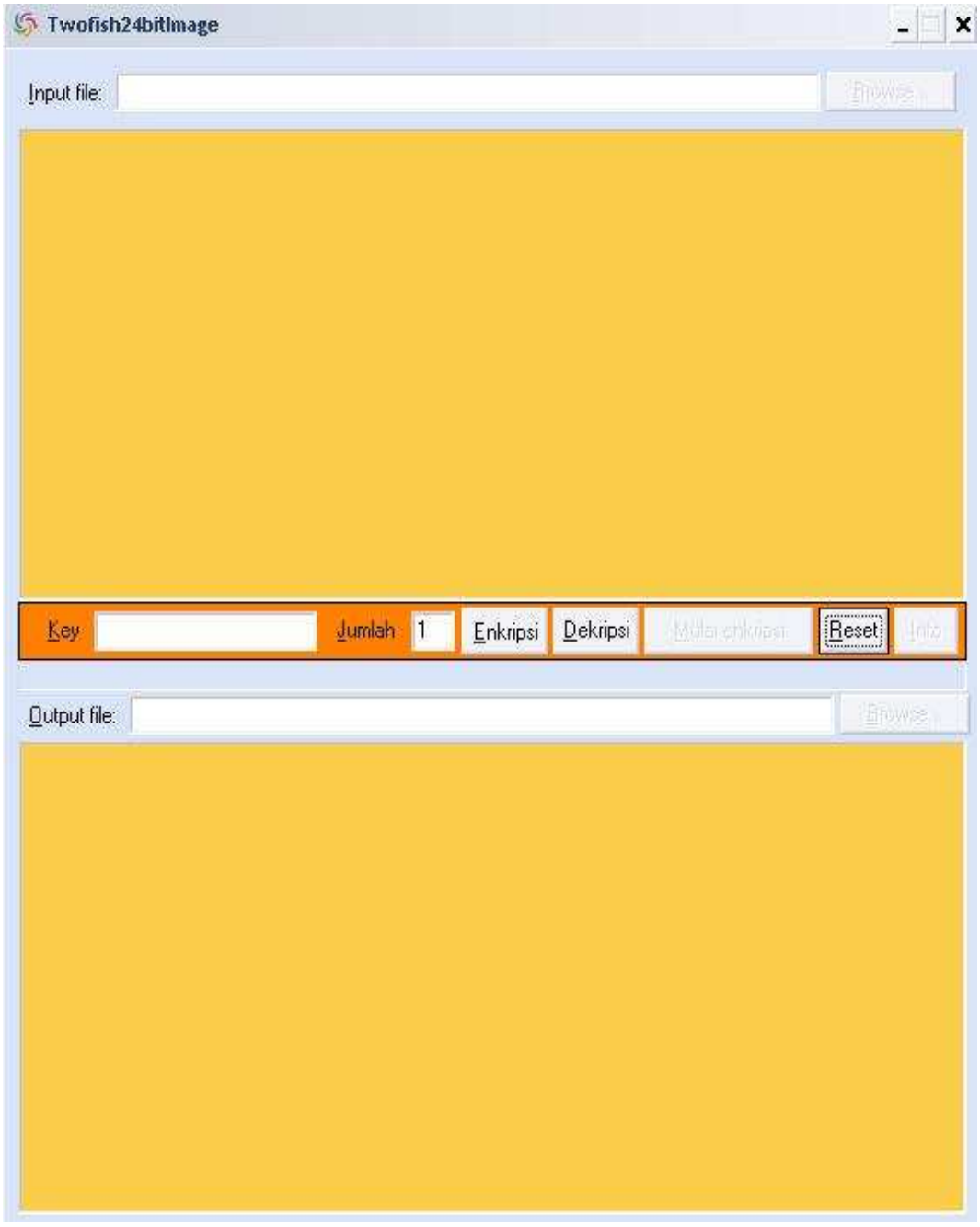
2. Perangkat Lunak

   Perangkat lunak dalam implementasi ini menggunakan:

   a. Sistem Operasi *Windows XP Professional*
   b. Bahasa pemrograman *Borland Delphi 7.0*

5.1.4 Tampilan aplikasi

Aplikasi pada tugas akhir ini dirancang untuk membantu mengamankan *file* gambar dengan format .bmp 24 bit. Tampilan antar muka (interface) dari aplikasi yang dirancang terlihat seperti gambar 5.1 dibawah ini.



Gambar 5.1 Tampilan Menu Utama Aplikasi Penyandian Citra

1. Tombol Enkripsi

   Digunakan untuk menampilkan tombol mulai enkripsi dan menonaktifkan tombol deskripsi agar tidak terjadi kesalahan proses.

2. Tombol Dekripsi

   Digunakan untuk menampilkan tombol mulai deskripsi dan menonaktifkan tombol enkripsi agar tidak terjadi kesalahan proses.

3. Tombol Reset

   Digunakan untuk menghapus semua inputan pada aplikasi untuk memulai kerja baru.

4. Tombol Info

   Digunakan untuk mengetahui ukuran / kapasitas *file* citra dan waktu yang digunakan untuk proses enkripsi atau dekripsi.

5. Input *file*

   Merupakan *file* asal dari objek yang akan dienkripsi / dekripsi yang dicari melalui tombol browse.

6. Output *file*

   Merupakan *file* tempat hasil enkripsi / dekripsi disimpan menggunakan tombol browse

7. Key

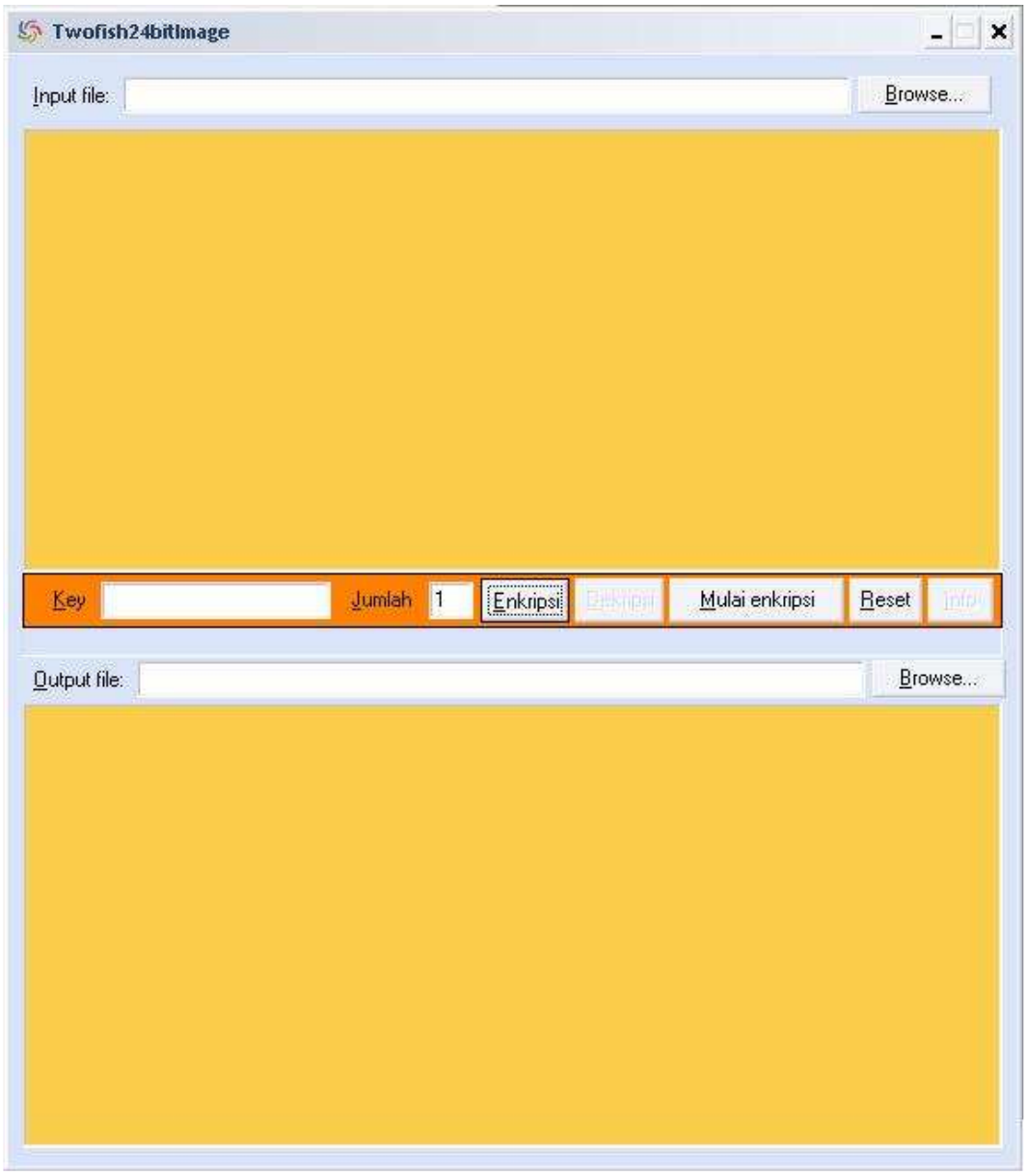
   Kunci atau password dari proses enkripsi / dekripsi

8. Jumlah

   Merupakan jumlah proses enkripsi / dekripsi yang digunakan.

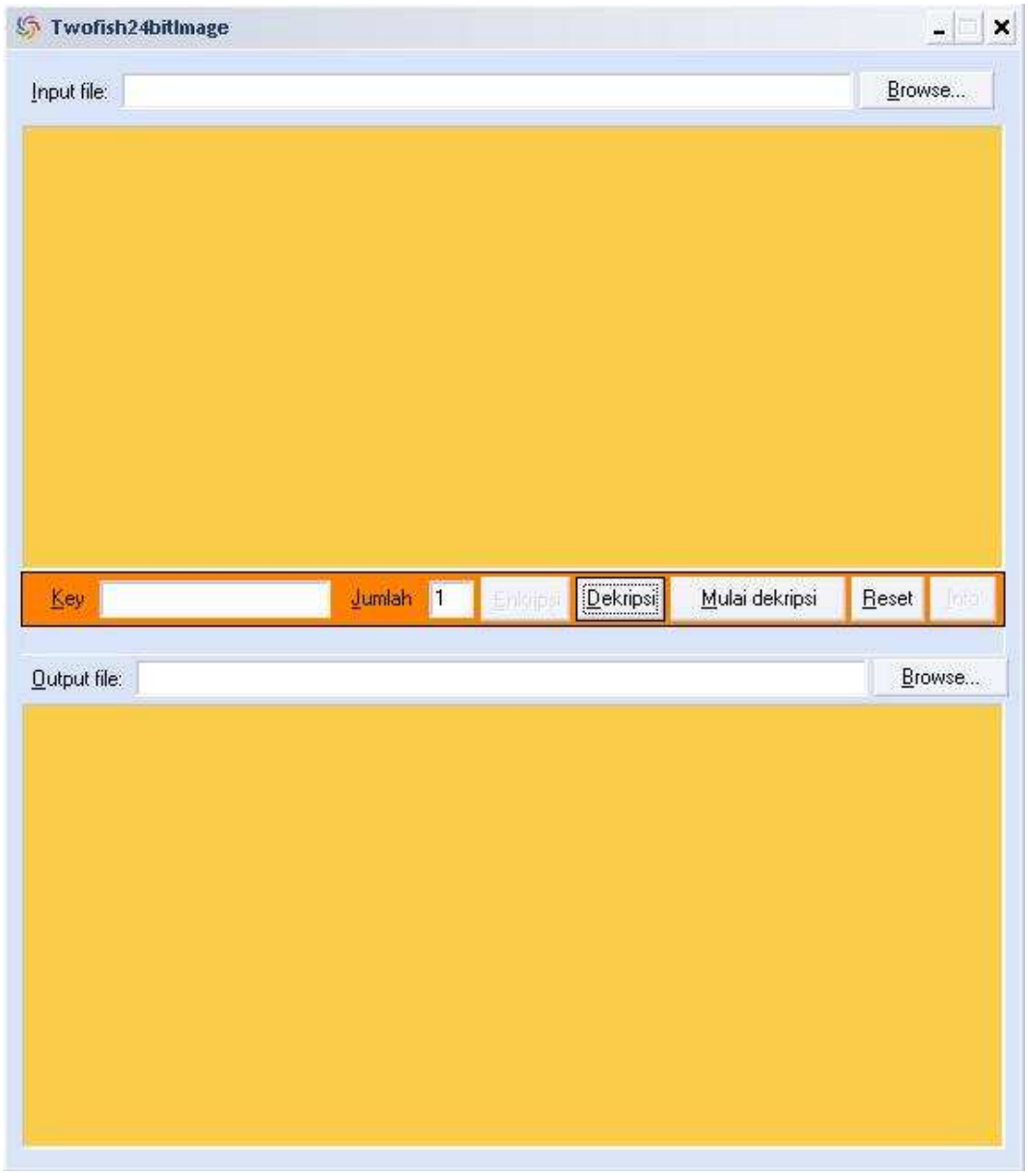
Pada menu tampilan diatas terlihat beberapa proses yang harus dijalankan secara berurutan sesuai dengan yang dijalaskan pada *flowchart* pada bab IV diatas.

Selanjutnya tampilan enkripsi, tombol mulai enkripsi, tombol browse menjadi aktif sedangkan tombol dekripsi menjadi tidak aktif. Terlihat seperti gambar 5.2 dibawah ini.



Gambar 5.2 Tampilan Menu Enkripsi

Tampilan berikutnya merupakan menu dekripsi pada gambar 5.3 terlihat tombol enkripsi tidak aktif dan tombol mulai dekripsi menjadi aktif, hal ini dirancang agar pengguna tidak melakukan kesalahan pada proses dekripsi.



Gambar 5.3 Tampilan Menu Dekripsi

5.2 Pengujian Sistem

Setelah tahap implementasi selesai dilakukan maka pengujian terhadap sistem perlu dilakukan. Hal ini berguna untuk mengetahui apakah masih terdapat *error* atau tidak.

5.2.1 Lingkungan Pengujian Sistem

Pengujian sistem ini dilakukan pada lingkungan perangkat lunak dan perangkat keras sesuai dengan lingkungan implementasi.

1. Pengujian terhadap fungsi-fungsi aplikasi

   Pengujian ini bertujuan untuk melihat apakah semua fungsi atau proses pada aplikasi dapat berjalan dengan baik, yaitu dengan menguji proses enkripsi dan dekripsi terhadap *file* citra bmp 24bit menggunakan algoritma *twofish* dan menguji semua prosses yang ada pada aplikasi.

2. Pengujian jumlah enkripsi dan dekripsi

   Pengujian ini bertujuan untuk melihat pengaruh jumlah proses enkripsi dan dekripsi terhadap lama proses enkripsi dan dekripsi. Jumlah enkripsi dan dekripsi ini juga bertujuan untuk pengamanan pada kriptografi dimana jumlah enkripsi harus sesuai dengan jumlah dekripsi.

3. Pengujian terhadap waktu

   Pengujian ini bertujuan untuk mengetahui lama proses enkripsi dan dekripsi, dimana waktu yang digunakan menentukan efektifitas dari sebuah aplikasi kriptografi. Pengujian terhadap waktu pada aplikasi penyandian citra menggunakan algoritma *Twofish* ini dapat dilihat pada tabel 5.9.

4. Pengujian terhadap *Attack* / serangan.

   Tujuan dari kriptografi adalah untuk pengamanan data jadi keamanan dari serangan / *Attack* adalah syarat mutlak dari sebuah aplikasi

kriptografi. Pada aplikasi ini pengujian terhadap *Attack* yang dilakukan adalah dengan metode *trial and error,* yang dapat dilihat pada tabel 5. 11.

Pengujian keberhasilan aplikasi pada tugas akhir ini bergantung pada sukses atau tidaknya aplikasi meng-enkripsi *file* citra bmp dan mengubahnya menjadi gambar yang tidak dapat dikenali serta mampu mengembalikan gambar seperti gambar awal.

5.2.2 Pengujian Tampilan Aplikasi

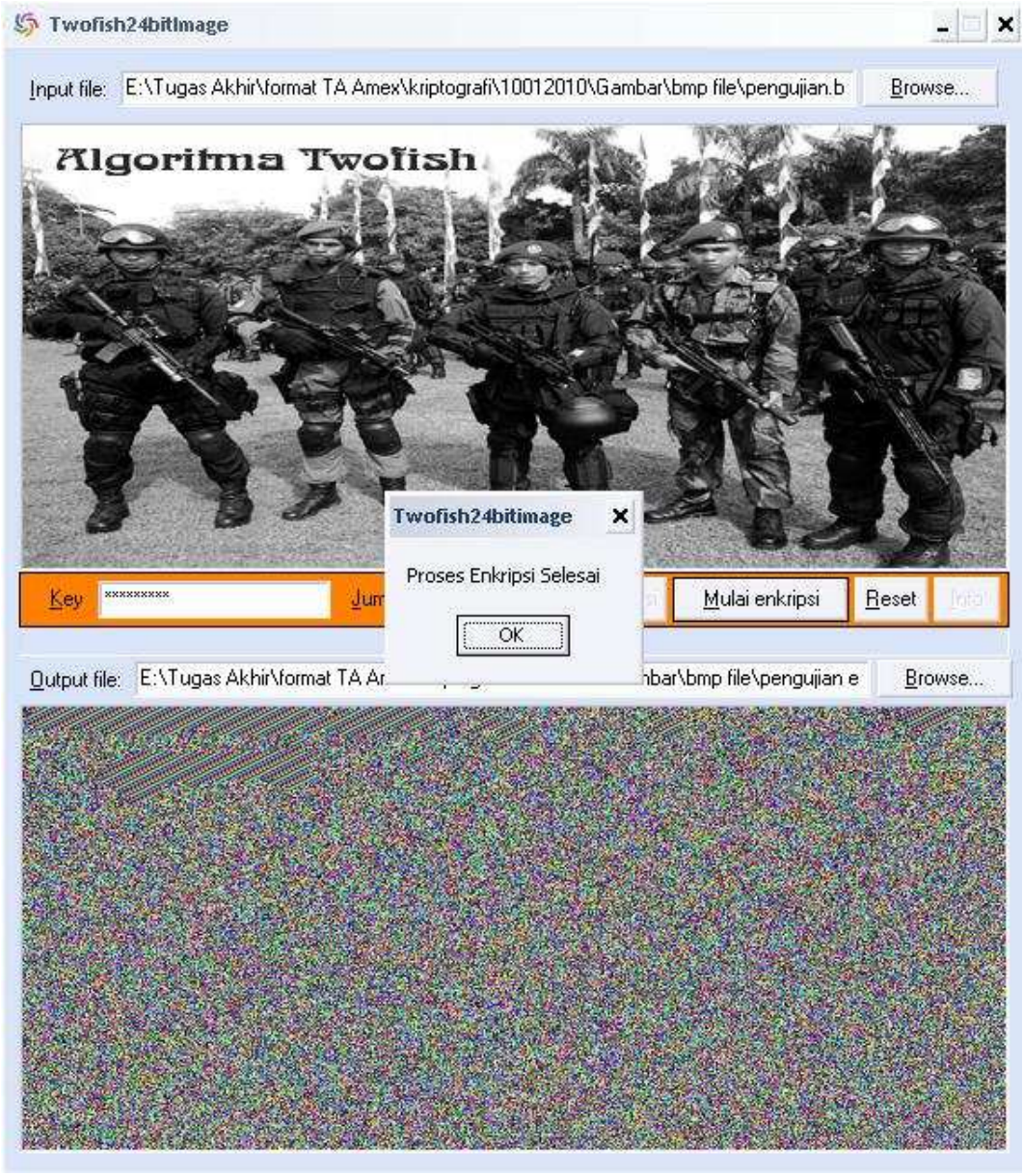
Pengujian dilakukan dengan melihat tampilan masing-masing menu yang ada dalam setiap urutan proses enkripsi maupun pada proses dekripsi

5.2.2.1 Pengujian Tampilan Proses Enkripsi.

Pada tampilan awal diperlihatkan *menu* pencarian gambar melalui tombol *browse* yang berfungsi untuk mencari gambar, tombol ini dapat di-*klik* hingga munculah tempat penyimpanan gambar seperti *directory* yang ada pada *windows*. Aplikasi akan serta merta memunculkan gambar bertipe bmp 24bit, jika *file* gambar tidak bertipe bmp 24bit maka aplikasi tidak mengenalinya atau tidak muncul pada *directory*. Setelah *file* gambar dimasukkan maka ditampilan atau *display* pada aplikasi akan terisi gambar yang kita masukan tadi, secara umum seperti yang terlihat pada gambar 5.4

Selanjutnya untuk melakukkan proses enkripsi kita juga harus menentukan tempat penyimpanan gambar hasil enkripsi, untuk melakukan uji coba akan diuji

dengan mengambil *file* **”Pengujian.bmp”** dan memasukkan kata kunci **”pengujian”** dengan jumlah proses enkripsi sebanyak 1 kali dan menyimpan *file* di *directory* yang sama dengan nama **”pengujian enkripsi”.** Secara umum proses diatas terlihat seperti pada gambar 5.4 dibawah ini. Setelah semua proses selesai dilakukan maka proses enkripsi dapat dijalankan.



Gambar 5.4 Proses Enkripsi

Setelah proses enkripsi selesai maka muncullah *massage dialog* yang menampilkan proses enkripsi selesai dijalankan. Tombol ok dapat di klik sehingga tombol mulai enkripsi menjadi tidak aktif, hal ini agar tidak terjadi proses enkripsi yang berulang. Untuk lebih jelas dapat dilihat gambar 5.5 dibawah ini.

Gambar 5.5 Proses Enkripsi Selesai

5.2.2.2 Pengujian Tampilan Proses Dekripsi.

Sama halnya seperti proses enkripsi, hanya saja pada proses dekripsi gambar yang kita masukkan adalah gambar hasil enkrip berupa gambar yang sudah tidak dapat dikenali lagi.

Untuk menguji aplikasi pada tugas akhir ini penulis mengambil hasil enkripsi yang telah dilakukan tadi, yaitu *file* gambar dengan nama **"pengujian enkripsi.bmp"** dengan kunci **"pengujian"** dan jumlah enkripsi **"1"** kali disimpan dengan nama **"gambar asli"** secara kesuluruhan proses ini dapat dilihat seperti gambar 5.6 dibawah ini.

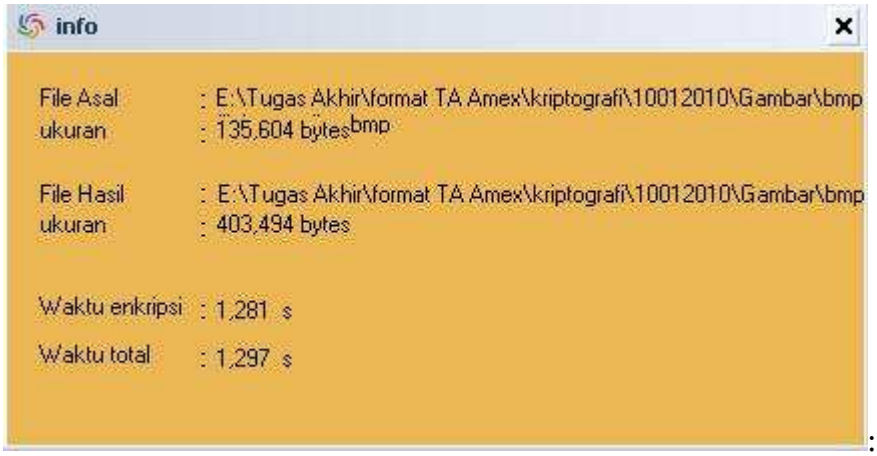
Gambar 5.6 Proses Dekripsi

Setelah proses diatas selesai, maka muncul *massage dialog* yang menyatakan proses dekripsi selesai dilakukan, setelah tombol ’ok’ diklik maka tombol mulai dekripsi menjadi tidak aktif hal ini berguna agar proses dekripsi tidak dilakukan berulang kali oleh pengguna. Secara umum proses ini dapat terlihat pada gambar 5.7 di bawah ini.

5.7 Proses Dekripsi selesai

5.2.2.3 Pengujian Tampilan Info.

Menu info berisi tentang informasi lokasi penyimpanan dan kapasitas *file* awal dan *file* hasil enkripsi atau dekripsi serta waktu yang dibutuhkan untuk kedua proses tersebut. Secara umum *menu* info tersebut telihat pada gambar 5.7 dibawah ini:



5.8 Tampilan *menu* Info

5.2.3 Pengujian Modul.

Pengujian modul merupkan hasil pengujian implementasi secara detail tentang cara kerja dan menjalankan aplikasi sehingga user dapat mempelajari aplikasi lewat modul pengujian ini.

5.2.3.1 Pengujian Modul Enkripsi.

Merupakan hasil pengujian implementasi aplikasi secara detail mengenai item-item yang terdapat pada setiap tampilan proses enkripsi citra bmp 24bit.

1 Pengujian Tahap 1 Proses Enkripsi *File*.

Prekondisi

1. Aplikasi dalam keadaan siap untuk digunakan, tombol yang aktif tombol enkripsi dan dekripsi.

Tabel 5.1 Tabel Butir Uji Pengujian Tahap 1 Proses Enkripsi *File*

Des Kripsi	Prekondisi	Prosedur Pengujian	Masu Kan	Keluaran yang Di harapkan	Kriteria Evaluasi Hasil	Hasil yang didapat	Kesim pulan
Pengujian tahap I proses enkripsi	Tombol mulai tidak aktif	1. tampil menu proses mulai enkripsi 2. Klik tombol ”enkripsi” Setelah proses berhasil tombol mulai enkripsi aktif. 3. Masukkan *file* yang akan dienkripsi, password dan jumlah enkripsi serta tempat penyimpanan hasil enkripsi	*File* citra tipe .bmp24 bit	Proses enkripsi berhasil, dalam proses ini tidak ada instruksi error	Proses enkripsi berhasil, dalam proses ini tidak ada instruksi error	Proses enkripsi berhasil, dalam proses ini tidak ada instruksi error	Di terima

2 Pengujian Tahap 2 Mencari Sumber *file* dan Tempat Penyimpanan *File* Citra

Prekondisi

1. Sudah ada sumber *file* citra didalam perangkat penyimpanan yang akan dilakukan pengujian.

Tabel 5.2 Tabel Butir Uji Pengujian Tahap 2 Mencari Sumber *File* dan Tempat Penyimpanan *File* Citra

Deskripsi	Prekondisi	Prosedur Pengujian	Masukan	Keluaran yang Diharapkan	Kriteria Evaluasi Hasil	Hasil yang didapat	Kesimpulan
Pengujian tahap 2 mencari sumber dan tempat penyimpanan *file* citra	Tampilan layar menu utama, dengan cara menekan tombol browse pada label *file* input dan *file* output	1. pada label *file* input tekan tombol browse di pojok kiri 2. cari *file* citra yang akan diuji 3. pada label *file* output tekan tombol browse di pojok kiri bawah 4. tentukan nama dan lokasi penyimpanan *file* yang akan di enkripsi	Sumber *file* citra	Proses pembacaan *file* berhasil, tidak ada instruksi error	Proses pembacaan *file* berhasil, tidak ada instruksi error	Proses pembacaan *file* berhasil, tidak ada instruksi error	Di terima

3. Pengujian Tahap 3 Maukkan kata kunci

Tabel 5.3 Tabel Butir Uji Pengujian Tahap 3 Memasukkan *key*.

Des Kripsi	Prekondisi	Prosedur Pengujian	Masukan	Keluaran yang Di harapkan	Kriteria Evaluasi Hasil	Hasil yang didapat	Kesimpulan
Pengujian tahap 3 mencari sumber *file*	Tahap 1 dan 2 telah dilalui tidak ada instruksi error	1. Tampil menu *key*	Bilangan berupa angka dengan jumlah maximal 15 karakter	Key berhasil diinputkan tidak ada instruksi error	Key berhasil diinputkan tidak ada instruksi error	Key berhasil diinputkan tidak ada instruksi error	Di terima

4 Pengujian Tahap 4 Jumlah Enkripsi.

Prekondisi

1. Tahap 1, 2 dan 3 telah dilalui tidak ada instruksi error

Tabel 5.4 Tabel Butir Uji Pengujian Tahap 3 Proses Enkripsi *File*

Des Kripsi	Prekondisi	Prosedur Pengujian	Masu Kan	Keluaran yang Di harapkan	Kriteria Evaluasi Hasil	Hasil yang didapat	Kesimpulan
Pengujian tahap 4 jumlah enkripsi	Tahap 1, 2 dan 3 telah dilalui tidak ada instruksi error	Isikan jumlah enkripsi	Bilangan berupa angka dengan jumlah minimal 0 dan maksimal 999	Jumlah enkripsi berhasil diinputkan tidak ada instruksi error	Jumlah enkripsi berhasil diinputkan tidak ada instruksi error	Jumlah enkripsi berhasil diinputkan tidak ada instruksi error	Di terima

5.2.3.2 Pengujian Modul Dekripsi.

Pengujian modul ini merupakan hasil pengujian implementasi aplikasi secara detail mengenai item-item yang terdapat pada setiap tampilan proses dekripsi.

1. Pengujian Tahap 1 Menentukan *File* Hasil Ekripsi.

Prekondisi

1. Secara umum tidak terdapat perbedaan antara *file* gambar asli dengan *file* gambar hasil enkripsi, hanya saja *file* hasil enkripsi biasanya sudah tidak dapat dikenali atau gambarnya telah rusak. Jadi prekondisi pada modul dekripsi ini adalah *file* hasil enkripsi telah tersimpan diperangkat penyimpanan yang akan dilakukan pengujian.

Tabel 5.5 Tabel Butir Uji Pengujian Menentukan *File* Hasil Enkripsi Yang Akan Didekripsi

Deskripsi	Prekondisi	Prosedur Pengujian	Masukan	Keluaran yang Diharapkan	Kriteria Evaluasi Hasil	Hasil yang didapat	Kesimpulan
Pengujian menentukan *file* hasil Enkripsi yang akan didekripsi	Sudah ada sumber *file* hasil enkripsi	1. Masukkan *file* hasi enkripsi	*File* hasil enkripsi	Muncul gambar hasil enkripsi pada tampilan aplikasi	Muncul gambar hasil enkripsi pada tampilan aplikas dan tidak ada instruksi error	Muncul gambar hasil enkripsi pada tampilan aplikas dan tidak ada instruksi error	Di terima

2 Pengujian Tahap 2 Memasukkan Kata Kunci.

. Prekondisi

1. Tahap 1 telah dilalui tidak ada instruksi error

Tabel 5.6 Tabel Butir Uji Pengujian Tahap 2 Memasukkan Kata Kunci

Deskripsi	Prekondisi	Prosedur Pengujian	Masukan	Keluaran yang Diharapkan	Kriteria Evaluasi Hasil	Hasil yang didapat	Kesimpulan
Pengujian Tahap 2 Memasukkan Kata Kunci	Tahap 1 telah dilalui tidak ada instruksi error	1. Tampil menu untuk memasukkan kata kunci 2. Silahkan ketik kata kuncinya sesuai kata kunci pada saat enkripsi	Kata kunci pada proses enkripsi	Jika kata kunci benar masuk keproses berikutnya jika salah hasil proses dekripsi tidak menemukan *file* asli dan tidak ada instruksi error.	Jika kata kunci benar masuk keproses berikutnya jika salah hasil proses dekripsi tidak menemukan *file* asli dan tidak ada instruksi error.	Jika kata kunci benar masuk keproses berikutnya jika salah hasil proses dekripsi tidak menemukan *file* asli dan tidak ada instruksi error.	Di terima

3 Pengujian Tahap 3 Jumlah Dekripsi.

Prekondisi

1. Tahap 1 dan 2 telah dilalui tidak ada instruksi error

Tabel 5.7 Tabel Butir Uji Pengujian Tahap 3 Proses Enkripsi *File*

Des Kripsi	Prekondisi	Prosedur Pengujian	Masu Kan	Keluaran yang Di harapkan	Kriteria Evaluasi Hasil	Hasil yang didapat	Kesimpulan
Pengujian tahap 3 proses dekripsi *file*	Tahap 1 dan 2 telah dilalui tidak ada instruksi error	1. Tampil menu jumlah dekripsi	Bilangan angka yang sama seperti proses enkripsi	Jika terdapat perbedaan antara proses enkripsi dan dekripsi maka aplikasi tetap menjalankan proses dekripsi namun tidak akan menemukan gambar yang asli dan tidak terjadi error	Jika terdapat perbedaan antara proses enkripsi dan dekripsi maka aplikasi tetap menjalankan proses dekripsi namun tidak akan menemukan gambar yang asli dan tidak terjadi error	Jika terdapat perbedaan antara proses enkripsi dan dekripsi maka aplikasi tetap menjalankan proses dekripsi namun tidak akan menemukan gambar yang asli dan tidak terjadi error	Di terima

4 Pengujian Tahap 3 Proses Dekripsi *File*.

Prekondisi

1. Tahap 1, 2 dan 3 telah dilalui tidak ada instruksi error

Tabel 5.8 Tabel Butir Uji Pengujian Tahap 4 Proses Dekripsi *File*

Des Kripsi	Prekon disi	Prosedur Pengujian	Masu Kan	Keluaran yang Di harapkan	Kriteria Evaluasi Hasil	Hasil yang didapat	Kesim pulan
Pengujia n tahap 4 proses dekripsi	Tahap 1, 2 dan 3 telah dilalui tidak ada instruksi error	1. tampil menu Dekripsi 2. Klik tombol ”ok” Setelah proses berhasil akan tampil pesan ”proses dekripsi selesai	*File* gambar hasil enkripsi	Proses enkripsi berhasil, dalam proses ini tidak ada instruksi error	Proses enkripsi berhasil, dalam proses ini tidak ada instruksi error	Proses enkripsi berhasil, dalam proses ini tidak ada instruksi error	Di terima

5.2.4 Pengujian Terhadap Jumlah Enkripsi dan Dekripsi

Jumlah enkripsi dan dekripsi untuk pengujian pada aplikasi ini haruslah sama, jika terdapat perbedaan maka gambar asli tidak akan ditemukan, tapi jumlah dekripsi tidak mutlak harus langsung sama dengan jumlah enkripsi ia dapat dilakukan enkripsi secara berulang hingga jumlah enkripsi sama dengan jumlah dekripsi. Misalnya kita enkripsi suatu *file* dengan jumlah 3 kali enkripsi,pada saat proses membalikkan bisa kita lakukan 3 kali proses dekripsi dengan syarat hasil proses dekripsi ke-1 dilakukan proses dekripsi kembali setelah itu hasil dekripsi ke-2 didekripsi-kan kembali maka hasil nya akan menampilkan gambar semula.

5.2.5 Pengujian Terhadap Waktu Dengan Jumlah Enkrpsi Yang Berdeda.

Pengujian terhadap waktu yang dibutuhkan dengan jumlah enkripsi yang berbeda untuk membuktikan perbedaan waktu yang dibutuhkan baik proses enkripsi maupun proses dekripsi.

Tabel 5.9 Pengujian terhadap jumlah ekripsi sebanyak 2 kali.

No	Nama *File*	Ukuran *File* (Bytes)	Kata kunci (Password)	Jumlah Enkripsi	Waktu Enkripsi	Waktu Dekripsi
1	tif.bmp	185,526 bytes	12345	2 kali	01.34 s	01.34 s
2	gradien.bmp	525,874 bytes	12345	2 kali	1.265s	1.25 s
3	window.bmp	3686.454 bytes	12345	2 kali	7.984s	7.968 s
4	no limit.bmp	153.718 bytes	12345	2 kali	2.078 s	2.002 s
5	1st.bmp	810.056 bytes	12345	2 kali	1.829s	1.859s

Tabel 5.10 Pengujian terhadap jumlah ekripsi sebanyak 20 kali.

No	Nama *File*	Ukuran *File* (Bytes)	Kata kunci (Password)	Jumlah Enkripsi	Waktu Enkripsi	Waktu Dekripsi
1	tif.bmp	185,526 bytes	12345	20 kali	3.047 s	3.031 s
2	gradien.bmp	525,874 bytes	12345	20 kali	8.3595s	8.422 s
3	window.bmp	3686.454 bytes	12345	20 kali	58.141s	58.735 s
4	no limit.bmp	153.718 bytes	12345	20 kali	14.627 s	14.766 s
5	1st.bmp	810.056 bytes	12345	20 kali	12.813s	12.937s

Dari analisa pengujian kedua tabel diatas dapat ditarik kesimpulan sebagai berikut:

1. Ukuran *file* dan banyaknya jumlah enkripsi mempengaruhi waktu dari proses enkripsi dan dekripsi.
2. Waktu yang dibutuhkan untuk proses dekripsi lebih lama bila dibandingkan dengan proses enkripsi namun, perbedaan antara keduanya tidak terlalu besar sehingga dapat diabaikan.

5.2.6 Pengujian terhadap *Attack* dengan cara mencoba memasukkan kemungkinan kunci

Pengujian kekuatan algoritma *twofish* dari serangan dengan menggunakan jenis serangan (*attack*). Percobaan yang dibuat untuk mengungkap *plaintext* atau

kunci dengan mencoba semua kemungkinan kunci (*trial and error*). Berikut merupakan cuplikan dari percobaan kunci yang salah dari sebuah *file*, yang telah dienkripsi terlebih dahulu.

Gambar 5.9 Proses Dekripsi dengan Kunci yang Salah

Aplikasi tetap akan menjalankan perintah dan mengeluarkan *output* berupa gambar tapi gambar yang dihasilkan tidak kembali seperti gambar yang asli. Hal

ini disebabkan aplikasi tidak mempunyai *database* untuk menyipan kunci dari proses enkripsi.

5.2.5.1 Batasan Pengujian Terhadap *Attack*

Batasan pengujian terhadap attack pada tugas akhir ini adalah sebagai berikut :

1. Pengujian dalam proses *attack* terhadap algoritma dilakukan sebanyak 20 kali percobaan.
2. *File* yang digunakan pada percobaan adalah *file* 131.bmp

Asumsi yang digunakan dalam pengujian terhadap *attack*:

1. Kriptanalis tidak mengetahui kunci dan memasukkan kunci secara acak

Tabel 5.11 Pengujian Terhadap *Attack*

No	Ukuran *file* *(bytes)*	*Password* awal	*Password* Uji coba	Hasil
1	2.559.030 *bytes*	*twofish*	123456	Tidak berhasil
2	2.559.030 *bytes*	*twofish*	asdfgt	Tidak berhasil
3	2.559.030 *bytes*	*twofish*	wer435	Tidak berhasil
4	2.559.030 *bytes*	*twofish*	asdqwe	Tidak berhasil
5	2.559.030 *bytes*	*twofish*	654321	Tidak berhasil
6	2.559.030 *bytes*	*twofish*	678905	Tidak berhasil
7	2.559.030 *bytes*	*twofish*	aszxcf	Tidak berhasil
8	2.559.030 *bytes*	*twofish*	bgfdrs	Tidak berhasil
9	2.559.030 *bytes*	*twofish*	gfgthy	Tidak berhasil
10	2.559.030 *bytes*	*twofish*	drfesw	Tidak berhasil
11	2.559.030 *bytes*	*twofish*	frtyuj	Tidak berhasil
12	2.559.030 *bytes*	*twofish*	sdweas	Tidak berhasil
13	2.559.030 *bytes*	*twofish*	asdefr	Tidak berhasil
14	2.559.030 *bytes*	*twofish*	asdrfg	Tidak berhasil
15	2.559.030 *bytes*	*twofish*	67yt5r	Tidak berhasil
16	2.559.030 *bytes*	*twofish*	wedrs1	Tidak berhasil
17	2.559.030 *bytes*	*twofish*	aqswed	Tidak berhasil
18	2.559.030 *bytes*	*twofish*	gtyhuj	Tidak berhasil
19	2.559.030 *bytes*	*twofish*	nmkloi	Tidak berhasil
20	2.559.030 *bytes*	*twofish*	hnjuyi	Tidak berhasil

Dari tabel diatas dapat disimpulkan bahwa algoritma *twofish* tidak memiliki kunci lemah. Sangat sulit untuk menembus algoritma ini dengan cara mencoba kemungkinan kunci.

5.3 Kesimpulan Pengujian

Adapun kesimpulan dari pengujian yang telah dilakukan adalah sebagai berikut:

1. Jumlah enkripsi dan dekripsi merupakan pengamanan sekunder setelah kata kunci atau *password*. Seseorang yang telah mendapatkan *chiperteks* dan kunci akan tetap kesulitan dalam mengkombinasikan jumlah dekripsinya.
2. Untuk pengamanan yang lebih komplek pengguna dapat menambahkan jumlah proses enkripsi yang lebih banyak, namun penambahan ini akan berpengaruh pada waktu proses enkripsi maupun proses dekripsi. Dimana semakin banyak jumlah enkripsi maka semakin besar waktu yang dibutuhkan untuk proses enkripsi dan dekripsi.
3. Aplikasi tidak dapat membalikkan gambar yang telah dienkripsi menjadi gambar aslinya apabila gambar hasil enkripsi tersebut di rotasi, karena nilai tinggi dan lebar *pixel* gambar tersebut telah berubah.

BAB VI

PENUTUP

6.1 Kesimpulan

Kesimpulan yang dapat diambil dari tugas akhir ini antara lain:

1. Aplikasi tidak mengamankan pesan dari penghapusan dan peng-*edit*-an sehingga apabila data dihapus atau di*edit,* pada saat proses dekripsi data tidak kembali ke gambar semula.
2. Kelebihan dari aplikasi yang dirancang pada tugas akhir ini ialah, aplikasi mempunyai pengamanan sekunder selain kunci yaitu jumlah enkripsi. sehingga diperlukan kombinasi yang cukup rumit apabila pihak ketiga mendapatkan *cipertext* dan kunci tanpa mengetahui berapa kali jumlah enkripsi dilakukan.
3. Kelemahan aplikasi yang dirancang pada tugas akhir ini ialah, aplikasi tidak menyimpan kunci sehingga, apabila *user* melakukan enkripsi secara berulang kemungkinan kehilangan *file* asli sangat besar karena kunci untuk dekripsi sudah tidak diketahui lagi.

6.2 Saran

Agar penulisan laporan penelitian tugas akhir ini bermanfaat dan berdaya guna dimasa sekarang dan yang akan datang, maka penulis memberikan beberapa saran sebagai berikut:

1. Aplikasi ini hanya terbatas pada citra dengan format *.bmp, untuk itu disarankan agar dapat dikembangkan dengan format lain agar ruang lingkup penggunaan algoritma *twofish* lebih luas.
2. Pada tugas akhir ini pengujian serangan (*attack*) hanya dilakukan dengan mencoba kemungkinan penggunaan kunci untuk itu agar dapat

dikembangkan motode serangan (*attack*) yang lainnnya guna menguji kekuatan algoritma *twofish*.

3. Diharapkan apabila ada penelitian lanjutan mengenai penyandian citra menggunakan algoritma *Twofish* aplikasi yang dirancang agar dapat mengatasi masalah peng*edit*an *file,* sehingga apabila ada sebuah file yang telah dienkripsi lalu dirotasi aplikasi tetap dapat membalikkan gambar menjadi gambar aslinya.

Daftar Pustaka

Anton, H., "*Aljabar Linear Elementer, Edisi Kelima"*, Jakarta, Erlangga, 1987.

Andi. *Memahami Model Enkripsi dan Security Data*, Andi Offset, Yogyakarta,2003.

Ariyus. D, *Pengantar Ilmu Kriptografi Teori, Analisis, dan Implementasi*, Andi Yogyakarta, Yogyakarta. 2008

Chodowiec.P *"Implementation of the Twofish Cipher Using FPGA Devices"* [online]Avalible.http://www.schneier.com/blog/archives/2005/11/Twofish_reprt .pdf/, diakses 1 Maret 2009.

Kristanto, K, *Keamanan Data Pada Jaringan Komputer*, Gaya Media, Klaten. 2003

Munir, Rinaldi. *Diktat Kuliah IF 5054 Kriptografi*. Institut Teknologi Bandung: Bandung.2006

Munir, Rinaldi. *Kriptgrafi*, Informatika. Bandung.2006

Rahardjo, Budi. *Keamanan system informasi berbasis internet*. Institut Teknologi Bandung: Bandung.1998

Setiadi, Budi. *analisis sistem keamanan data dengan menggunakan metode des dan metode gost.* Institut Teknologi Bandung: Bandung.2004

Wikipedia." *Cryptograpy*" [online]:

Avalible. http://en.wikipedia.org/wiki/Cryptograpy, diakses 12 Januari 2010